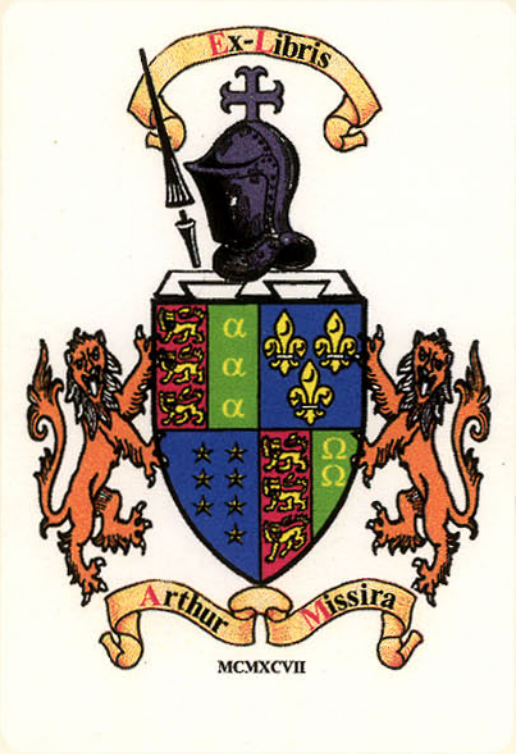
Computers in Medicine Series

General Editor: D. W. HILL

Digital Filters

MARTIN H. ACKROYD

Butterworths



£1·95
net
in UK only

Computers in Medicine Series

under the General Editorship of

D. W. HILL

M.Sc., Ph.D., F. Inst. P., F.I.E.E.
Reader in Medical Physics, Royal
College of Surgeons of England

Digital Filters

MARTIN H. ACKROYD

B.Sc.,Ph.D.
Lecturer in Signal Processing,
University of Technology,
Loughborough,
Leicestershire

London : Butterworths

ENGLAND: BUTTERWORTH & CO. (PUBLISHERS) LTD.
LONDON: 88 Kingsway, WC2B 6AB

AUSTRALIA: BUTTERWORTHS PTY. LTD.
SYDNEY: 586 Pacific Highway, 2067
MELBOURNE: 343 Little Collins Street, 3000
BRISBANE: 240 Queen Street, 4000

CANADA: BUTTERWORTH & CO. (CANADA) LTD.
TORONTO: 14 Curity Avenue, 374

NEW ZEALAND: BUTTERWORTHS OF NEW ZEALAND LTD.
WELLINGTON: 26–28 Waring Taylor Street, 1

SOUTH AFRICA: BUTTERWORTH & CO. (SOUTH AFRICA) (PTY) LTD.
DURBAN: 152–154 Gale Street

Suggested U.D.C. Number: 681·325·022

ISBN 0 407 52700 1

Printed in Great Britain by
Redwood Press Limited
Trowbridge, Wiltshire

Contents

Editor's Foreword

The aim of this *Computers in Medicine* series is to present an account of both the clinical applications and the computer science aspects of computing in medicine. The balance between these two will naturally vary from volume to volume, but each author will write from personal experience of his specialty. Medical computing is advancing so rapidly that it can best be described by a coherent series of individual monographs.

Digital filter techniques are of considerable importance in the processing of physiological data by digital computers. Dr. Ackroyd has set down the essentials of digital filter design in this monograph in a concise fashion which brings out his skill as a teacher. Whilst digital filters have a special place in off-line signal processing on large machines, they are also being increasingly used in smaller computers for on-line applications. Much of the key literature on digital filters is scattered around the communications journals which are not generally accessible to bio-medical personnel. Dr. Ackroyd has provided a valuable service by bringing sufficient material into a monograph and by supplying a set of tested Fortran IV subroutines for the design and evaluation of filters.

D. W. H.

Chapter 1

Introductory Concepts

In modern methods of medical research, diagnosis and patient monitoring, physiological variables are frequently recorded or processed as electrical signals. The signals may be obtained directly in electrical form as in the case of electroencephalogram (EEG) or electrocardiogram (ECG) signals, or they may be obtained from a transducer, as in the case of blood pressure or flow measurements. Frequently, it is desirable or even essential to filter such signals prior to subsequent interpretation or analysis.

Signals may be contaminated by interference or noise of some origin such as, for example, 50 Hz mains hum. Interference can often be removed or at least attenuated by an appropriate filtering process. The experimenter may perhaps wish to eliminate all frequencies outside some spectral band prior to further analysis. He may wish to search for certain features in the waveform. Many such operations can be accomplished by the use of a filter. Until recently, the term *filtering* automatically implied the use of an electrical network. However, it is also possible to perform filtering by numerical operations which can be performed by a digital computer.

This book is concerned with filtering operations carried out within a digital computer. A *digital filter*, in broad terms, is any device which accepts a sequence of numbers as its input and operates on them to produce another number sequence as its output. Digital filters can be constructed as special purpose hardware units and, indeed, such units are now commercially available. However, throughout this book a digital filter will be understood to be implemented as a program which is run on a general purpose digital computer.

Digital filters implemented on a computer have several advantages over the analogue filter networks that might alternatively be used. Some of the advantages of digital filters are the following.

Accuracy

The inaccuracies of digital filters, which are due to the rounding errors in the computer arithmetic, can be made as small as required. The background noise produced in analogue circuits cannot be so easily controlled and analogue components cannot easily be made to a tolerance of less than about 1 per cent.

Versatility

A computer can easily be programmed to implement a digital filter with characteristics so complicated that they could not be produced, for practical reasons, by an equivalent analogue filter. The alteration of a digital filter design involves, at most, the re-writing of a section of program code or often merely the reading-in of a different set of filter coefficients as data.

Freedom from Drift

The characteristics of a digital computer program remain the same each time it is re-run, irrespective of variations in mains supply voltage, ambient temperature, and so on. This is an important advantage in medical applications where very low frequencies are often involved and the effects of drift in analogue components can appear as spurious signals.

Most frequently, digital filters are used simply because a computer is available and there is filtering to be done. Often, the availability of a small laboratory computer will make it possible to use digital filters in applications where the expense of purpose-built analogue filters cannot be justified; or a system which already incorporates a computer, such as a patient monitoring system, may need to perform some kind of filtering as part of its operation.

DIGITIZATION

Before a signal that exists as a wiggly waveform can be processed by a digital computer, it must be converted to a sequence of numbers, a process which is called digitization or analogue-to-digital (A/D) conversion. An analogue-to-digital converter is usually incorporated as part of the computer system. It works so that at successive, equally-spaced instants of time, such as every millisecond, for example, it measures the value of its input voltage. This *sampling* process is normally done under the control of a clock which may be part of the computer hardware. Each time the analogue-to-digital converter samples

the input voltage it produces a number which represents the input voltage at that instant and which forms the input to the computer.

The numbers produced by an analogue-to-digital converter are normally either fixed-point binary or binary-coded decimal, according to its design. In either case, there is a limit to the magnitude of the numbers that it can output. If the input voltage exceeds the value represented by this largest possible output number, overload occurs, and there is gross discrepancy between the input voltage level and the output number. The amplitude of the input signal is normally adjusted so that overload seldom occurs.

Error is introduced in the analogue-to-digital conversion process as a result of the finite number of digits in the output numbers. Each possible output number represents a particular voltage level, and an input voltage lying somewhere between two such quantization levels produces an output number which indicates the nearest quantization level. In the case of an analogue-to-digital converter producing binary numbers of N bits (including the sign bit), the difference between adjacent quantization levels is $\frac{1}{2}^N$ of the difference between the positive and negative overload levels. Thus, for example, a 10-bit analogue-to-digital converter, with maximum and minimum input voltages of 5·12 and −5·12 V, will have a 10 mV interval between quantization levels.

The error that results from rounding each input sample to the nearest quantization level produces an effect which is much the same as adding random noise to the input voltage. The root-mean-square value of this equivalent random noise is $1/\sqrt{12}$ of the difference between adjacent quantization levels.

Table 1 shows the signal-to-noise ratios obtainable from analogue-to-digital converters with various numbers of binary digits. There are many

TABLE 1

Number of bits, including sign bit	*Signal-to-noise ratio (dB)*
6	38
7	44
8	50
9	56
10	62
11	68
12	74
13	80
14	86

ways of defining signal-to-noise ratio, but here it is defined as the ratio of the root-mean-square value of the largest amplitude sine wave that can be handled without overload to the root-mean-square value of the quantization noise. This definition of signal-to-noise ratio is commonly used in amplifier and tape recorder specifications.

Table 1 shows, for example, that if the signal to be digitized is obtained from a tape machine with a signal-to-noise ratio of 50 dB, then there is little to be gained by using an analogue-to-digital converter of more than nine bits. This assumes, of course, that the signal amplitude is adjusted to make full use of the input voltage range of the converter.

Choosing a suitable sampling rate is often something of a problem as there are conflicting requirements involved. If the sampling rate is high, the signal can be reconstructed easily from its samples, whereas if the sampling rate is low, more computation can be done between samples and more elaborate digital filters can be used in 'real time'.

The sampling theorem of communication theory states that a waveform can, in principle, be reconstructed from its sample values without any degree of approximation – provided that the sampling rate exceeds twice the frequency above which the spectrum of the signal is zero. However, in practice the sampling theorem does not provide much help. There are two reasons for this. First, physical signals do not have spectra which are zero above some particular frequency; the spectrum of a physical signal tails off above some frequency, but there is no frequency above which it is everywhere strictly zero. The second point is that if the output of the digital filter is to be reconstructed as a waveform, then a digital-to-analogue converter, possibly followed by a simple analogue filter, will be used rather than the ideal scheme demanded by the sampling theorem. For example, the waveform may be approximately reconstructed by joining adjacent sample points with straight lines. If the output is needed for visual presentation, the experimenter can judge for himself on the evidence of his eyes what sampling rate provides a visually acceptable picture. As a general guide, if the sampling rate is so high that it is possible to 'see' the waveform from a display of the sampled points, then it is almost certain that the sampling rate is high enough for any other purpose such as, for example, spectrum analysis. With such a sampling rate, the clock frequency is usually five to ten times the nominal bandwidth of the signal.

One important point to note is that the sampling rate is determined not by the highest frequency which is of interest but by the effective bandwidth of the signal. Thus, if a signal whose highest significant

frequency component lies in excess of 100 Hz is to be analysed, it must be sampled at a rate in excess of 200 Hz, even though only frequency components below 10 Hz are of interest.

NUMBER SEQUENCES AND Z-TRANSFORMS

As a digital filter operates not on waveforms but on sequences of numbers it is necessary to introduce a notation to describe such sequences. In this book a sequence of numbers is represented by writing its elements in order between brackets like this: $(a_0, a_1, a_2, \ldots)$. The three stops following a_2 indicate that there may be other non-zero elements in the sequence which have not been shown. If all the elements after a_M are zero, this can be indicated by writing $(a_0, a_1, \ldots, a_M)$.

Each element of a sequence is associated with a particular instant of time, the *clock instant* or, synonymously, the *sampling instant,* at which it is generated by the analogue-to-digital converter. The converter is assumed to operate at equally spaced instants of time and the spacing between clock instants, the clock period or sampling interval, is denoted by T. Thus, the element a_0 is associated with time $t = 0$, the element a_1 is associated with time $t = T$, a_2 with time $t = 2T$, and so on.

In the analysis of digital filters the *z-transform* plays a part analogous to the role of the Laplace transform in analogue filter theory. Fortunately, the theory of the z-transform is, on the whole, simpler than that of the Laplace transform. The z-transform of the sequence $(a_0, a_1, a_2, \ldots)$, is denoted by the symbol A(z). It is a series in powers of z^{-1}:

$$A(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots$$

z is actually a complex variable, as will be seen later, but often it is just manipulated as an algebraic symbol and it is unnecessary to bother about its precise nature. To invert a z-transform, that is, to find the sequence $(a_0, a_1, \ldots)$ corresponding to A(z), is very simple when the the z-transform is already expanded in the form of a series. The sequence element with subscript k is simply the coefficient of z^{-k} in the series.

Example

The sequence (1, 1, 1, −1, 1) has the z-transform

$$1 + z^{-1} + z^{-2} - z^{-3} + z^{-4}$$

Example

The z-transform $1/(1 - \frac{1}{2}z^{-1})$ can be expanded, by use of the binomial theorem, into the form

$$1 + \tfrac{1}{2}z^{-1} + \tfrac{1}{4}z^{-2} + \tfrac{1}{8}z^{-3} + \ldots$$

The sequence of which this is the z-transform is thus

$$(1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \ldots).$$

Lower case letters are normally reserved to denote sequences and the corresponding capital letters to denote their z-transforms.

DIGITAL FILTERS: BLOCK DIAGRAMS

A computer will be understood to be working as a digital filter if, at regular clock instants equally spaced by an interval T, it accepts a number as its input and produces a number as its output. Each output number is to be computed using only the three following computational operations:

(1) Storage of numbers.
(2) Multiplication of numbers by constant coefficients.
(3) Addition of numbers.

These operations are assumed to take place instantaneously at the clock instants. Even though a digital filter will eventually be implemented as a computer program, it is helpful to describe the computational processes involved by the use of *block diagrams* in which the three permitted operations are represented as symbols.

The operation of storage of numbers is represented by a rectangular block, as shown in *Figure 1(a)*. A number is applied to the input of

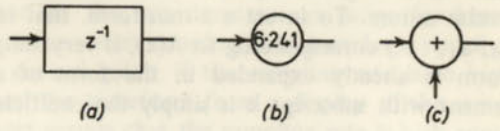


Figure 1. Block diagram symbols: (a) *one clock period storage;* (b) *multiplication by constant coefficient;* (c) *addition*

such a storage block at a clock instant. This number is then held by the block for one clock interval and appears at its output an infinitesimal time before the next number is applied to its input. It is assumed that,

initially, all the storage blocks in a digital filter hold zeros, so that when a digital filter is 'switched on' a zero appears at the output of each storage block just as the first number is applied to its input.

If the number sequence applied at the input to a storage block is $(a_0, a_1, a_2, \ldots)$, whose z-transform is $A(z) = a_0 + a_1z^{-1} + a_2z^{-2} + \ldots$, then the output sequence will be $(0, a_0, a_1, a_2, \ldots)$, whose z-transform is $a_0z^{-1} + a_1z^{-2} + a_2z^{-3} + \ldots = z^{-1}\,A(z)$. The important conclusion here is that the z-transform of the sequence at the output of a storage unit is the z-transform of its input sequence multiplied by z^{-1}.

Multiplication by a constant coefficient is represented by a symbol such as that shown in *Figure 1(b)*. At each clock instant a number is applied to the input of such a unit and instantaneously the product of the input number and the coefficient appears at the output.

Addition is represented by the symbol shown in *Figure 1(c)*. At every clock instant numbers are applied to the inputs of the addition unit and their sum instantaneously appears at the output.

In a practical digital filter the various operations cannot be performed instantaneously. For example, the addition of two numbers in a computer typically takes several microseconds. Nor, for that matter, can several arithmetic operations be performed simultaneously. Nevertheless, provided that all the operations needed can be completed within one filter clock period, this is normally of no account. *Figure 2* shows a very simple digital filter. If a sequence $(1, 0, 0, 0, \ldots)$, is applied at its input, what will be the output sequence? With such a simple filter, the output sequence is easily calculated directly.

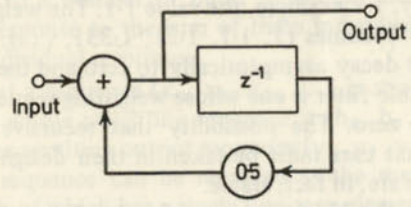


Figure 2. Simple recursive digital filter

At the initial clock instant the following events occur: the zero which is initially held in the storage block appears at its output, and is multiplied by 0·5. This product is added to the first element of the input sequence, which is 1 in this case. The sum appears at the filter output and is also applied as the input to the storage block. Thus, the output from the filter at the initial clock instant is 1. This is the first element in the output sequence.

At the following clock instant, a similar set of events occurs. This time the output from the storage unit is 1, while the input to the filter is a zero. As a result, the output from the digital filter at the second clock instant is 0·5. At the third clock instant it is 0·25, and so on. The output sequence in this case is (1, 0·5, 0·25, 0·125, . . .); it is the digital equivalent of a decaying exponential waveform.

The output sequence that a digital filter produces when its input sequence is (1, 0, 0, 0, . . .), plays such an important part in digital filter theory that it is given the special name of the *weighting sequence.* It is also often called the *impulse response,* despite the fact that digital filters accept numbers as their inputs and not waveforms or impulses.

RECURSIVE AND NON-RECURSIVE FILTERS

It is convenient to divide digital filters into two classes: those that are recursive and those that are non-recursive.

A recursive filter is one whose block diagram contains one or more closed paths. Expressed another way, a recursive filter is one which incorporates feedback. The block diagram of *Figure 2* contains a closed path and is thus the block diagram of a recursive filter. This example illustrates one property that only a recursive filter may possess – its weighting sequence may be of infinite extent. The weighting sequence of the filter of *Figure 2,* (1, ½, ¼, ⅛, . . .), dies away but it never completely reaches zero.

Another property that a recursive filter may have is that of *instability.* This can be illustrated by replacing the coefficient 0·5 in the filter of *Figure 2* by, for example, the value 1·1. The weighting sequence of the filter then becomes (1, 1·1, 1·21, 1·331, . . .). The weighting sequence does not decay asymptotically to zero and the filter is said to be unstable. A stable filter is one whose weighting sequence does decay asymptotically to zero. The possibility that recursive filters can be unstable means that care must be taken in their design to ensure that the resulting filters are, in fact, stable.

Figure 3 shows the block diagram of a simple, non-recursive filter; there is no closed path to be found in this block diagram. An important property of all non-recursive digital filters is that their weighting sequences are of finite length. For the filter of *Figure 3,* the weighting sequence is (1/16, 1/4, 3/8, 1/4, 1/16). There is no possibility of a non-recursive filter being unstable for, being of finite length, the weighting sequence reaches zero after a finite number of clock instants.

The design methods for recursive and non-recursive filters are quite different in nature, and are dealt with separately in Chapters 2 and 3.

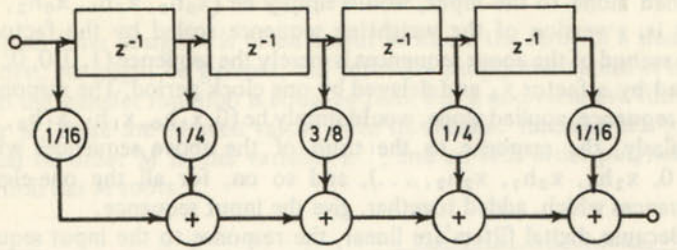


Figure 3. Simple non-recursive filter

TRANSFER FUNCTIONS

The section on block diagrams, page 6, illustrated how it is possible to calculate directly the output sequence of a digital filter given its input sequence. The technique used in that section was to duplicate the numerical operations that would be used by the filter itself. While this method suffices to give the output from a particular filter in response to a specific input, it does not give much information about the general properties of the filter. An alternative and more illuminating way of calculating the output from a digital filter is by the use of z-transforms.

Digital filters, which make use only of the computational operations of storage, multiplication by constant coefficients and addition, have the property of *linearity.* This means that it is possible to work out the responses to a number of input sequences individually and then calculate the response to the sum of these individual inputs by adding together the individual responses.

Suppose that a sequence $(x_0, x_1, x_2, \ldots)$, is applied as the input to a digital filter whose weighting sequence is $(h_0, h_1, h_2, \ldots)$. How do we calculate the resulting output sequence $(y_0, y_1, y_2, \ldots)$?

The input sequence can be regarded as the sum of the following sequences, each of which has a single non-zero element:

$(x_0, 0, 0, 0, \ldots)$

$(0, x_1, 0, 0, \ldots)$

$(0, 0, x_2, 0, \ldots)$ etc.

The first of these sequences is merely the sequence $(1, 0, 0, 0, \ldots)$, scaled by a factor x_0. The response to this first sequence, if it were

applied alone to the input, would simply be $(x_0h_0, x_0h_1, x_0h_2, \ldots)$, that is, a version of the weighting sequence scaled by the factor x_0. The second of the above sequences is merely the sequence $(1, 0, 0, 0, \ldots)$, scaled by a factor x_1 and delayed by one clock period. The response to this sequence, applied alone, would simply be $(0, x_1h_0, x_1h_1, x_1h_2, \ldots)$. Similarly, the response to the third of the above sequences will be $(0, 0, x_2h_0, x_2h_1, x_2h_2, \ldots)$, and so on, for all the one-element sequences which, added together, give the input sequence.

Because digital filters are linear, the response to the input sequence $(x_0, x_1, x_2, \ldots)$, is the sum of these individual response sequences. The z-transform $Y(z)$ of the filter output sequence is thus the sum of the z-transforms of these individual response sequences:

$$Y(z) = x_0 H(z) + x_1 z^{-1} H(z) + x_2 z^{-2} H(z) + \ldots$$
$$= X(z) H(z)$$

where $H(z)$ is the z-transform of the weighting sequence and $X(z)$ is the z-transform of the filter input sequence. This important result says that the z-transform of the output sequence of a digital filter is simply the product of the z-transforms of the input sequence and the weighting sequence. The z-transform of the weighting sequence is called the *transfer function* of the digital filter. The terms *z-transfer function* and *pulse transfer function* are also sometimes used.

With a non-recursive filter, the weighting sequence contains only a finite number of non-zero elements. The transfer function of a non-recursive filter is thus a polynomial in the variable z^{-1}:

$$H(z) = h_0 + h_1 z^{-1} + \ldots + h_M z^{-M}$$

The transfer function of a recursive filter is more complicated. It can, in fact, always be expressed as the ratio of two polynomials in z^{-1}:

$$H(z) = \frac{a_0 + a_1 z^{-1} + \ldots + a_M z^{-M}}{1 + b_1 z^{-1} + \ldots + b_N z^{-N}}$$

The *order* of a transfer function is the value of M or N, according to which is the larger. The order of a transfer function is, in fact, equal to the minimum number of storage blocks that are needed to construct the block diagram of a digital filter which has that transfer function. A high order transfer function thus requires more computer memory in its implementation than one of low order, since each storage block corresponds to a computer memory element in the final filter.

POLES AND ZEROS

In digital filter design it is often useful to know the *zeros* of a transfer function. As might be guessed, the zeros are simply the values of z for which the transfer function is equal to zero. For a non-recursive filter of order M, there are M such values since its transfer function is a polynomial of order M in the variable z^{-1}, and an Mth order polynomial equation has M roots.

Example

A non-recursive filter with the weighting sequence (¼, ½, ¼) is sometimes known as a *Hanning* filter and can be used to effect a moderate degree of smoothing on noisy data. Its transfer function is

$$H(z) = ¼ + ½z^{-1} + ¼z^{-2} .$$

By equating $H(z)$ to zero and solving the resulting quadratic equation it is easily found that this transfer function has two zeros at $z = -1$.

The *poles* of a transfer function are the real or complex values of z for which it becomes infinite. In the case of a non-recursive filter transfer function the only such value is $z = 0$. However, with a recursive filter, the transfer function becomes infinite for each value of z at which the denominator polynomial $1 + b_1 z^{-1} + \ldots + b_N z^{-N}$ becomes zero. The poles and zeros can be displayed on a plot showing the complex z-plane. Poles or zeros at $z = 0$ are often omitted in such plots.

Example

Find the poles and zeros of the recursive filter transfer function

$$H(z) = \frac{1 + z^{-1}}{1 + z^{-2}}$$

The numerator polynomial $1 + z^{-1}$ is zero when $z = -1$. The denominator polynomial $1 + z^{-2}$ is zero when $z^2 = -1$ or when $z = \pm j$. The transfer function thus has a zero at $z = -1$ and poles at $z = \pm j$, as plotted in *Figure 4.*

If the zeros, $\alpha_1, \alpha_2, \ldots, \alpha_M$, and the poles, $\beta_1, \beta_2, \ldots, \beta_N$, of a transfer function are known, then it can be expressed in factored form as

$$H(z) = \frac{a_0 (1 - \alpha_1 z^{-1})(1 - \alpha_2 z^{-1}) \ldots (1 - \alpha_M z^{-1})}{(1 - \beta_1 z^{-1})(1 - \beta_2 z^{-1}) \ldots (1 - \beta_N z^{-1})}$$

The factored form of the transfer function is particularly useful when a block diagram is to be constructed in the serial form as described in the following section.

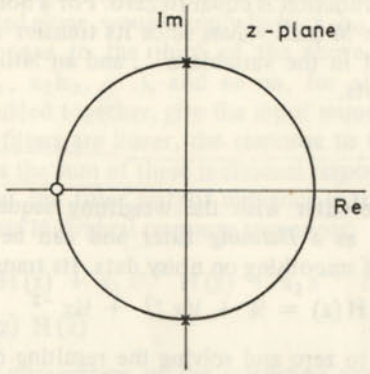


Figure 4. Pole-zero plot of transfer function $(1 + z^{-1})/(1 + z^{-2})$

The nature of the weighting sequence of a digital filter depends on the locations of its poles and zeros in the z-plane as described, for example, by Lindorff (1965). Usually, the exact form of the weighting sequence is not of very much interest if one is designing a filter to achieve a given frequency response. However, one very important property of the weighting sequence is whether or not it converges to zero; that is, whether or not the filter is stable. The stability of a digital filter is related to the position of the poles of its transfer function in the following way: if all the poles of the transfer function lie within the 'unit circle' in the z-plane, i.e. if they are less than unity in magnitude, the filter is stable. If one or more poles lies outside the unit circle, the filter is unstable. This is because the weighting sequence of a digital filter consists of the sum of a number of contributions with each pole providing one such contribution to the weighting sequence. A pole outside the unit circle provides a contribution which grows from one clock instant to the next while a pole inside the unit circle produces a contribution to the weighting sequence which dies away. The weighting sequence itself will decay only if all of these contributions die away. Thus, for the weighting sequence to die away to zero, all of the poles must lie within the unit circle. The zeros can lie anywhere. Whether they lie inside or outside the unit circle, they do not affect the stability of the filter.

It is essential that any filter which is to be used in practice should be stable, so that any design method for recursive filters must be sure to give poles which lie within the unit circle. The design methods presented in the next chapter start by choosing the poles so that they all lie within the unit circle. These methods are thus guaranteed to give stable filter designs.

BLOCK DIAGRAM CONSTRUCTION

When a suitable transfer function has been chosen for a particular purpose it remains to program a computer to implement the filter. This task is greatly facilitated by the construction of a suitable block diagram as an intermediate stage.

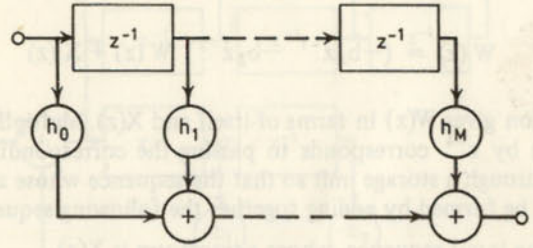


Figure 5. Block diagram of non-recursive filter with weighting sequence $(h_0, h_1, \ldots, h_M)$

With non-recursive filter transfer functions, there is little difficulty. It can readily be verified that the block diagram shown in *Figure 5* has the weighting sequence $(h_0, h_1, \ldots, h_M)$ and thus implements the transfer function.

$$H(z) = h_0 + h_1 z^{-1} + \ldots + h_M z^{-M}$$

With recursive filters it is not quite so obvious how to produce a suitable block diagram. One form of block diagram construction is called the *direct form*. In illustration of the technique it will be assumed that the third order transfer function

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

is to be programmed. The z-transform of the output of the filter can be expressed

$$Y(z) = H(z)\ X(z)$$

$$= (a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3})\ W(z) \qquad (1.1)$$

where $$W(z) = \frac{X(z)}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

The first step is to construct a block diagram of a digital filter whose output has the z-transform W(z) when its input has the z-transform X(z). This can be done by rearranging equation (1.1):

$$W(z) = (-b_1 z^{-1} - b_2 z^{-2})\ W(z) + X(z)$$

This equation gives W(z) in terms of itself and X(z). Multiplication of a z-transform by z^{-1} corresponds to passing the corresponding number sequence through a storage unit so that the sequence whose z-transform is W(z) can be formed by adding together the following sequences:

(1) The input sequence, whose z-transform is X(z).
(2) The sequence $(w_0, w_1, w_2, \ldots)$ delayed by one clock period and multiplied by $-b_1$.
(3) The same sequence delayed by two clock periods and multiplied by $-b_2$.

All this is accomplished by the block diagram shown in *Figure 6(a)*.

It now remains to form the output sequence $(y_0, y_1, y_2, \ldots)$ by

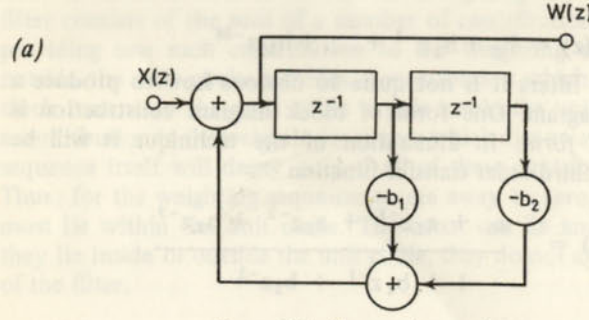


Figure 6(a). For caption see facing page

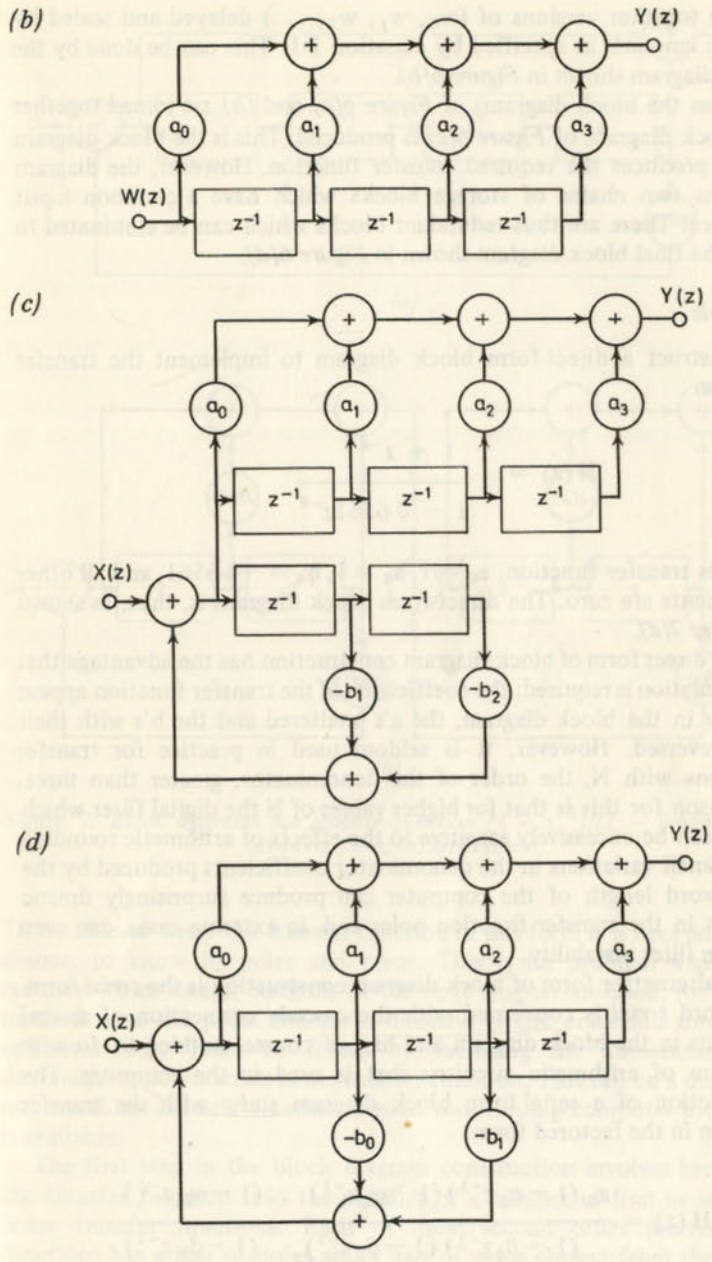


Figure 6. Stages in construction of block diagram in direct form

adding together versions of $(w_0, w_1, w_2, \ldots)$ delayed and scaled by various amounts as specified by equation 1·1. This can be done by the block diagram shown in *Figure 6(b)*.

When the block diagrams of *Figure 6(a)* and *(b)* are joined together the block diagram of *Figure 6(c)* is produced. This is the block diagram which produces the required transfer function. However, the diagram contains two chains of storage blocks which have a common input sequence. There are thus redundant blocks which can be eliminated to yield the final block diagram shown in *Figure 6(d)*.

Example

Construct a direct form block diagram to implement the transfer function

$$H(z) = \frac{1 + z^{-4}}{1 - 0{\cdot}6561z^{-4}}$$

For this transfer function, $a_0 = 1$, $a_4 = 1$, $b_4 = -0{\cdot}6561$, and all other coefficients are zero. The direct form block diagram is, thus, as shown in *Figure 7(a)*.

The direct form of block diagram construction has the advantage that no calculation is required; the coefficients of the transfer function appear directly in the block diagram, the a's unaltered and the b's with their signs reversed. However, it is seldom used in practice for transfer functions with N, the order of the denominator, greater than three. The reason for this is that for higher values of N the digital filter which results can be excessively sensitive to the effects of arithmetic rounding error. Small variations in the denominator coefficients produced by the finite word length of the computer can produce surprisingly drastic changes in the transfer function poles and, in extreme cases, can even result in filter instability.

An alternative form of block diagram construction is the *serial* form. The word *serial* is concerned with the cascade connection of several sub units in the block diagram and has, of course, nothing to do with the form of arithmetic circuitry that is used in the computer. The construction of a serial form block diagram starts with the transfer function in the factored form

$$H(z) = \frac{a_0 (1-\alpha_1 z^{-1})(1-\alpha_2 z^{-1}) \ldots (1-\alpha_M z^{-1})}{(1-\beta_1 z^{-1})(1-\beta_2 z^{-1}) \ldots (1-\beta_N z^{-1})}$$

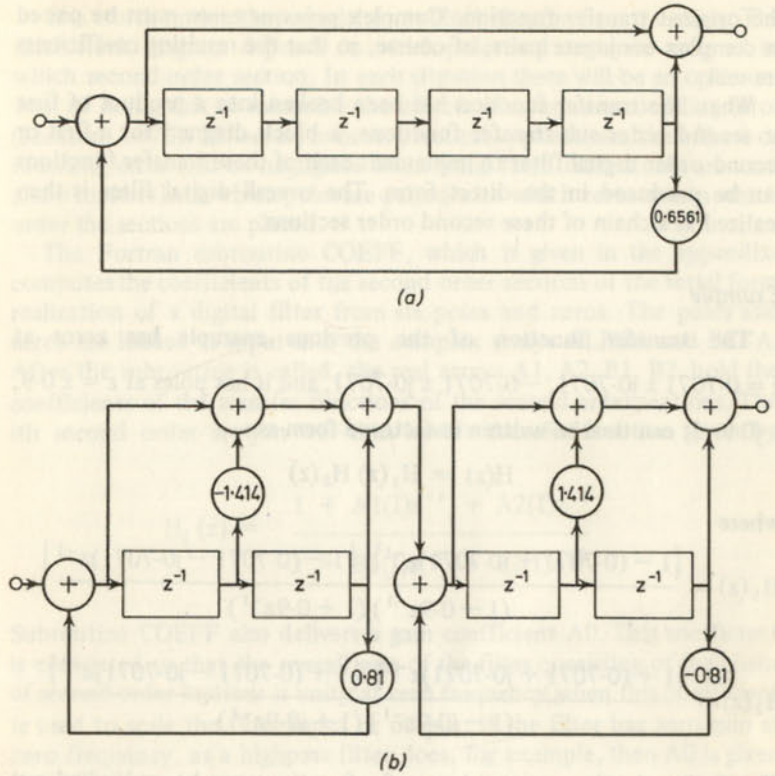


Figure 7. Block diagrams for transfer function $(1 + z^{-4})/(1 - 0{\cdot}6561z^{-4})$: (a) direct form; (b) serial form

To be able to write the transfer function in this form it is necessary, of course, to know its poles and zeros. This is no problem when the recursive filter design methods of the next chapter are used, for these methods give the poles and zeros directly. If the poles and zeros are unknown they must be found by factoring the numerator and denominator polynomials in the transfer function. This can be a difficult numerical operation even when a good root-finding computer program is available.

The first step in the block diagram construction involves breaking the transfer function into the product of a number of first or second order transfer functions. Each of these second order sub-transfer functions has a pair of poles and a pair of zeros chosen from those of

the original transfer function. Complex poles or zeros must be paired in complex conjugate pairs, of course, so that the resulting coefficients are real.

When the transfer function has been broken into a product of first or second order sub-transfer functions, a block diagram for a first or second order digital filter to implement each of these transfer functions can be produced in the direct form. The overall digital filter is then realized as a chain of these second order sections.

Example

The transfer function of the previous example has zeros at $z = 0{\cdot}7071 \pm j0{\cdot}7071, -0{\cdot}7071 \pm j0{\cdot}7071$, and it has poles at $z = \pm 0{\cdot}9, \pm j0{\cdot}9$. It can thus be written in factored form as

$$H(z) = H_1(z)\, H_2(z)$$

where

$$H_1(z) = \frac{\{1 - (0{\cdot}7071 + j0{\cdot}7071)z^{-1}\}\ \{1 - (0{\cdot}7071 - j0{\cdot}7071)z^{-1}\}}{(1 - 0{\cdot}9z^{-1})(1 + 0{\cdot}9z^{-1})}$$

$$H_2(z) = \frac{\{1 + (0{\cdot}7071 + j0{\cdot}7071)z^{-1}\}\ \{1 + (0{\cdot}7071 - j0{\cdot}7071)z^{-1}\}}{(1 - j0{\cdot}9z^{-1})(1 + j0{\cdot}9z^{-1})}$$

The factors in these two sub-transfer functions can be multiplied out to give

$$H_1(z) = \frac{1 - 1{\cdot}414z^{-1} + z^{-2}}{1 - 0{\cdot}81z^{-2}}$$

and

$$H_2(z) = \frac{1 + 1{\cdot}414z^{-1} + z^{-2}}{1 + 0{\cdot}81z^{-2}}$$

Figure 7(b) shows a serial form block diagram implementation of H(z) produced by cascading two block diagrams which are direct form implementations of $H_1(z)$ and $H_2(z)$.

It is worth noting that a serial form block diagram is not unique. The final block diagram depends on which poles and zeros are assigned to which second order section. In each situation there will be an optimum way of doing this to minimize the effects of computer rounding error (Jackson, 1970). However, in most practical applications, the effects of rounding error will be negligible when serial form realizations are used and it matters little which poles are paired with which zeros and in which order the sections are placed.

The Fortran subroutine COEFF, which is given in the appendix, computes the coefficients of the second order sections of the serial form realization of a digital filter from its poles and zeros. The poles and zeros are loaded as input into the complex arrays ALPHA and BETA. After the subroutine is called, the real arrays A1, A2, B1, B2, hold the coefficients of the transfer functions of the second order sections. The ith second order section will thus have a transfer function given by

$$H_I(z) = \frac{1 + A1(I)z^{-1} + A2(I)z^{-2}}{1 + B1(I)z^{-1} + B2(I)z^{-2}}$$

Subroutine COEFF also delivers a gain coefficient A0. This coefficient is computed so that the overall gain of the filter consisting of the chain of second order sections is unity at zero frequency when this coefficient is used to scale the filter input or output. If the filter has zero gain at zero frequency, as a highpass filter does, for example, then A0 is given the value unity.

SINUSOIDAL SEQUENCES: FREQUENCY RESPONSE

The reader may well be familiar with the concept of *frequency response*, as applied to analogue systems such as filters, transducers, tape recorders, and so on. The idea of frequency response applies equally well to digital filters even though one is concerned here with number sequences rather than with waveforms.

A *sinusoidal sequence* is simply a sequence of numbers that could have been produced by sampling a sinusoidal waveform at regularly spaced instants of time. The effect of digital filtering on such sequences is particularly simple. If the input to a digital filter is a sinusoidal sequence the output will also be such a sequence; the only way in which the input and output sequences may differ is in their amplitude

and phase. The frequency response of a filter is, loosely, the way in which the difference in amplitude and phase between input and output sequences depends upon the frequency of the input. Before discussing the notion of frequency response in more detail it is necessary to explain the exact meaning of the concepts of amplitude, phase and frequency when applied to number sequences rather than to waveforms.

The amplitude of a sinusoidal sequence is defined as the amplitude, or peak-value, of a sinusoidal waveform which yields the sequence when sampled. It is worth noting that the amplitude of a sinusoidal sequence, as given by this definition, may be appreciably greater than the largest number which occurs in the sequence. For example, the waveform $\cos(\pi t/2 - \pi/4)$ has unit amplitude yet if it is sampled at $t = 0, \pm1, \pm2, \ldots$ the sequence $(\ldots, 0{\cdot}707, 0{\cdot}707, -0{\cdot}707, -0{\cdot}707, \ldots)$ is produced, whose largest element is 0·707. Nevertheless, the amplitude of this sequence is taken as unity. A mere change in phase so that the waveform becomes $\cos(\pi t/2)$, results in the sample sequence becoming $(\ldots 1, 0, -1, 0, 1, \ldots)$, whose largest element is 1. The definition of amplitude that is used here has an advantage in that it ensures that the amplitude and phase of a sinusoidal sequence are not interdependent quantities. It also gives consistent results in other ways. For example, the root-mean-square value of both of the sequences above is 0·707 – the same as the root-mean-square value of the sampled waveform.

The phase and frequency of a sinusoidal sequence are defined in a similar manner as being the phase and frequency of a sinusoidal waveform which, when sampled, yields the sequence.

It can be seen that the frequency of a sinusoidal sequence is an ambiguous quantity since sine waves of differing frequencies can yield the same sequence when sampled at a given rate. For example, a zero frequency cosine wave (that is, a constant) of unit amplitude yields the sequence $(\ldots 1, 1, 1, \ldots)$ when sampled at $t = 0, \pm1, \pm2, \ldots$. Precisely the same sequence can be obtained by sampling a cosine wave of frequency 1 Hz at these instants, or one of 2 Hz, and so on. No distinction can be made between sinusoidal sequences whose frequencies differ by some integral multiple of the sampling, or clock, frequency. In fact, any sinusoidal sequence can be regarded as being produced by sampling a sine wave whose frequency lies somewhere from zero to half the clock rate. For this reason, it is sufficient to specify the characteristics of a digital filter over the range of frequencies from zero to half the sampling frequency or the *Nyquist frequency*, as it is known. In this book, all frequency response graphs cover this range, the frequency scale being marked in fractions of the clock frequency. In most applications of digital filtering the sampling rate is chosen so that the Nyquist frequency exceeds the highest frequency at which the signal contains appreciable energy. No ambiguity can then arise as it is known beforehand that the frequency of any sinusoidal component of the original waveform lies in the range of frequencies from zero to 1/2T.

As mentioned previously, if the input to a digital filter is a sinusoidal sequence, its output will also be such a sequence which can differ from the input in amplitude and phase only. This can be seen by considering that adding two sine waves of the same frequency just produces another sine wave of the same frequency – and a digital filter produces its output by adding together delayed and scaled versions of its input.

The ratio of the amplitude of the output sequence to the input amplitude is the *gain* of a digital filter. It may be expressed directly as a ratio or in decibels as twenty times the logarithm of the ratio. The gain characteristic of a digital filter is the way in which its gain varies with the frequency of its input sequence.

The phase shift of a digital filter is the difference between the phases of the output sequence and the input sequence. In practical work the phase shift is usually measured in degrees. The *phase characteristic* is the way in which the phase shift varies with frequency. It is to be emphasized that the concepts of gain and phase shift are normally only meaningful when the input signal is sinusoidal.

The gain and phase characteristics of a digital filter can be found from its transfer function. This is done by replacing z in its transfer function by $\exp(j2\pi fT)$, where f is the frequency in hertz and T is the sampling period in seconds. The resulting function of f, $H(e^{j2\pi fT})$, is called the frequency response function. The magnitude of the frequency response function gives the gain characteristic of the digital filter, and its argument or angle gives the phase characteristic.

Example

What are the gain and phase characteristics of the Hanning filter (page 11)?

The transfer function of the Hanning filter is $\frac{1}{4} + \frac{1}{2}z^{-1} + \frac{1}{4}z^{-2}$, so its frequency response function is

$$\begin{aligned} H(e^{j2\pi fT}) &= \tfrac{1}{4} + \tfrac{1}{2}e^{-j2\pi fT} + \tfrac{1}{4}e^{-j4\pi fT} \\ &= e^{-j2\pi fT}\,(\tfrac{1}{4}e^{j2\pi fT} + \tfrac{1}{2} + \tfrac{1}{4}e^{-j2\pi fT}) \\ &= e^{-j2\pi fT}\,\tfrac{1}{2}(1 + \cos 2\pi fT). \end{aligned}$$

Its gain characteristic is thus

$$|\,H(e^{j2\pi fT})\,| = \tfrac{1}{2}(1 + \cos 2\pi fT)$$

and its phase characteristic is

$$\arg H(e^{j2\pi fT}) = -2\pi fT.$$

The gain of the Hanning filter is unity at zero frequency and it gently falls to zero at the Nyquist frequency. Its phase shift is zero at zero frequency and when the frequency approaches half the sampling rate the output lags 180° behind the input.

The function $\exp(j2\pi fT)$ has unit magnitude for all values of frequency but its argument varies between 0 and 180° as the frequency is increased from zero to the Nyquist frequency, 1/2T. The frequency response function is, in fact, the transfer function evaluated on the unit circle in the z-plane – that is, at values of z for which $|z| = 1$. Each frequency corresponds to a specific point on the unit circle. For example, the value of the frequency response function at zero frequency is the value of the transfer function at $z = 1$ and the value of the frequency response function at the Nyquist frequency is the value of the transfer function at $z = -1$. In general, the frequency response function at frequency f is the value of the transfer function on the unit circle at angle $2\pi fT$ radians from the positive real axis.

The reason why replacing z in the transfer function by $\exp(j2\pi fT)$ gives the frequency response is, briefly, as follows. Instead of directly evaluating the output sequence when the input is a cosine sequence of frequency f, it is possible to evaluate the output when the input is the complex sequence $(\ldots, \exp(-j2\pi fT), 1, \exp(j2\pi fT), \ldots)$, of which the real part is a cosine sequence. Of course, it is not usually feasible, in practice, to apply a complex input to a digital filter because it will normally be programmed to accept only real numbers. Nevertheless, from the mathematical point of view, there is nothing wrong with applying a complex input. When such a complex exponential sequence is passed through a storage unit the output sequence produced is simply the input multiplied by $\exp(-j2\pi fT)$. Thus, for this special input sequence a delay of one clock period has the same effect on the sequence as a multiplication by $\exp(-j2\pi fT)$. As a result, the output sequence from the filter can be calculated by merely multiplying the complex exponential input sequence by $H(e^{j2\pi fT})$.

The real part of this output sequence, which is the response to the real part of the input, is just a cosine whose amplitude is $|H(e^{j2\pi fT})|$ and whose phase is $\arg H(e^{j2\pi fT})$. Since the real part of the input has unit amplitude and zero phase, these are just the gain and phase shift of the filter at frequency f.

A subroutine is given in the appendix for evaluating the gain and phase characteristics of recursive digital filters. As its input, subroutine FREQ accepts arrays which contain the coefficients of the second order transfer functions in a serial form realization of a digital filter. For example, it will directly accept as input the output from subroutine

COEFF. Subroutine FREQ delivers as its output two arrays which contain, respectively, the gain of the filter in decibels and the phase shift in degrees at a specified number of frequencies equally spaced from zero to the Nyquist frequency, inclusive. Another subroutine GAINPH, is used in conjunction with subroutine FFT for computing the frequency responses of non-recursive filters. Chapter 3 explains how subroutine FFT is used to compute the frequency response function of a non-recursive filter from its weighting sequence. Chapter 3 also explains how subroutine GAINPH is used to evaluate the filter gain, in decibels, and phase in degrees from the values of the frequency response function computed by FFT.

In many applications filters are required to have gain and phase characteristics which vary relatively gently with frequency such as in providing equalization for recording/playback systems. In other applications, however, filters are required which would ideally pass sinusoids of all frequencies within some frequency range without change in amplitude and which would completely attenuate sinusoids lying outside that frequency range. Such ideal filter gain characteristics are sometimes

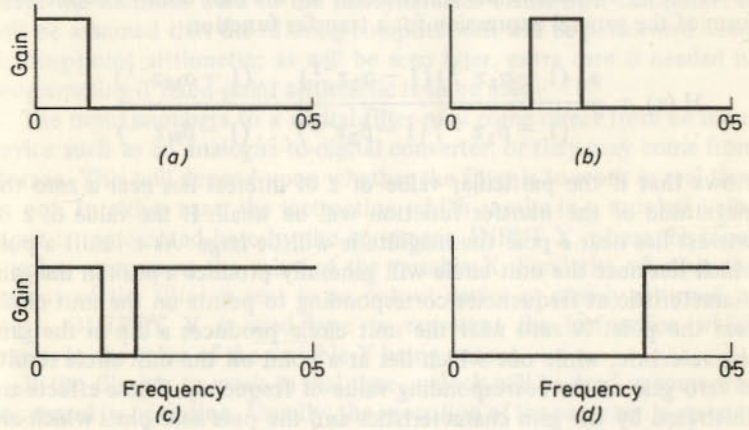


Figure 8. Ideal filter characteristics: (a) *lowpass;* (b) *bandpass;* (c) *bandstop;* (d) *highpass*

called *brickwall* characteristics. The four simplest ideal filter characteristics are the *lowpass, bandpass, bandstop* and *highpass* gain characteristics which are shown in *Figure 8.* Other more elaborate filter characteristics can be made by cascading filters of these types. For example, if it were required to reject the frequency components of a

signal in narrow bands near 50 and 100 Hz but to leave other frequency components unaltered, two bandstop filters with stopbands centred at 50 and 100 Hz could be cascaded to give an overall gain characteristic flat except for notches at these frequencies.

For ideal brickwall filter characteristics the *stopbands* are the frequency ranges over which the gain is zero and the *passbands* are the remaining frequency ranges. For a practical filter the passband and stopband are not so clearly defined. The passband is normally taken to be the frequency range over which the gain does not fall below some predetermined fraction of its maximum value, often 0·707 (−3 dB). The stopband is the range of frequencies over which the attenuation (the reciprocal of the gain), exceeds some other predetermined value, such as perhaps, 60 dB. The region that, with a practical filter, exists between the passband and the stopband is usually called the *transition band.*

If the pole-zero pattern of a transfer function is known, it is possible to sketch or visualize the frequency response without actually doing any calculations. As stated earlier, the frequency response function is obtained by evaluating the z-transform on the unit circle. The factored form of the general expression for a transfer function

$$H(z) = \frac{a_0 (1-\alpha_1 z^{-1})(1-\alpha_2 z^{-1}) \ldots (1-\alpha_M z^{-1})}{(1-\beta_1 z^{-1})(1-\beta_2 z^{-1}) \ldots (1-\beta_N z^{-1})}$$

shows that if the particular value of z of interest lies near a zero the magnitude of the transfer function will be small. If the value of z of interest lies near a pole the magnitude will be large. As a result a pole which lies near the unit circle will generally produce a peak in the gain characteristic at frequencies corresponding to points on the unit circle near the pole. A zero near the unit circle produces a dip in the gain characteristic, while one which lies at a point *on* the unit circle results in zero gain at the corresponding value of frequency. These effects are illustrated by the gain characteristics and the pole zero plots which are given in the next chapter.

PROGRAMMING DIGITAL FILTERS

When a digital filter design has been produced in the form of a block diagram the need to program it for a computer still remains. It is only possible here to give a general outline of the procedures that can be used since much of the detail will depend upon the characteristics of the individual computer on which the filter is to be implemented.

Up to this point it has been assumed that all the arithmetic operations portrayed in a block diagram occur simultaneously and instantaneously. A normal computer can, of course, only perform one arithmetic operation at a time, and each operation takes a finite interval of time. All of the arithmetic operations which have been assumed to occur simultaneously at each clock instant must, in fact, be completed in the period between clock instants. To achieve a high sampling rate the computer must be programmed to work efficiently. The speed of operation of a small computer is usually highest when a low-level or assembly language is used and when integer arithmetic is employed. Digital filter programs to work in real time are therefore often written in assembly language and use integer arithmetic. Programs to perform digital filtering off-line in a large computer on the other hand are often written in a high-level language such as Fortran, with floating point arithmetic.

It is sufficient to illustrate the principles involved in digital filter programming by the use of Fortran-like statements. The reader who needs to program digital filters in assembly language will be able to adapt the methods used to the idiosyncracies of his own computer. It will be assumed that the filtering computations will be performed using floating-point arithmetic; as will be seen later, extra care is needed in programming if fixed-point arithmetic is to be used.

The input numbers to a digital filter may come direct from an input device such as an analogue-to-digital converter, or they may come from storage. This will depend upon whether the filter is to work in real-time or not. In either case, the instruction which results in a number being input is represented here by the statement, INPUT X, where the input number appears as the value of the variable X. Similarly, whether the output of the filter is sent to an output device or merely returned to store, OUTPUT Y is used here to represent the instruction which results in the value of the variable Y being output.

If the filter is to work in real time, a clock will be used in some way to control its operation. Usually, the execution of the program is arrested at some point in the code until a clock signal arrives. The details here will depend upon the particular computer that is to be used. On the other hand, if a stored signal is being processed off-line, the computer can proceed at its own speed and the problems of clocking are irrelevent. The time scale in this case is determined by the sampling rate that was used when the stored data was originally generated.

The central idea in programming digital filters is that each storage block in the block diagram is made to correspond to a Fortran variable or, in assembly language programming, each storage block corresponds to a computer memory location. For example, in the fourth order filter

shown in *Figure 7(a)*, the contents of the first block can be the Fortran variable X1, the contents of the second X2, the third X3 and the fourth X4.

The first step in programming the filter is to form the new value that X1 will take at the clock instant. This can be done by the statement

```
HOLD  = X + 0.6051*X4
```

The new output number can now be formed by using the code

```
Y = HOLD + X4
```

After Y has been output it remains to replace the value of X4 by that of X3, the value of X3 by that of X2 and the value of X2 by that of X1. The value of X1 is then replaced by the value of HOLD. This is all done by the statements.

```
X4 = X3
X3 = X2
X2 = X1
X1 = HOLD
```

These are all the operations that need to be done in one clock interval. The complete code to implement the filter is thus

```
1 INPUT X
  HOLD = X + 0.6051*X4
  Y = HOLD + X4
  X4 = X3
  X3 = X2
  X2 = X1
  X1 = HOLD
  GO TO 1
```

If the filter is to operate in real-time, instructions must be included to ensure that it waits at some point in the program until the clock instant. The details of the programming to accomplish this depend entirely upon the way the particular computer and its real-time clock are organized.

A more complicated recursive filter consisting of a series of second order sections can be programmed by cascading sections of code similar to the above but with the output number produced by one block forming the input to the next. For filters of high order, it can be more

convenient to implement a second order section as a subroutine with the filter coefficients held in arrays rather than programmed as constants. Each section of the filter is then implemented as a call to the subroutine. For example, the following subroutine will implement a second order section.

```
SUBROUTINE SECTION (A1, A2, B1, B2, X1, X2, X, Y, N, I)
DIMENSION A1 (N), A2(N), B1(N), B2(N), X1(N), X2(N)
HOLD = X - B1(I) * X1(I) - B2(I) * X2(I)
Y = HOLD + A1(I) * X1(I) + A2(I) * X2(I)
X2(I) = X1(I)
X1(I) = HOLD
RETURN
END
```

The numerator coefficients of the transfer function of the ith filter section are held in the element I of the arrays A1 and A2, and the denominator coefficients are held in the element I of the arrays B1 and B2. The contents of the storage blocks in the ith filter section are held in element I of the arrays X1 and X2.

A tenth order filter using this subroutine could be implemented by the following code.

```
1 INPUT X
  DO 2 I = 1, 10
2 CALL SECTION (A1, A2, B1, B2, X1, X2, X, Y, N, I)
  OUTPUT Y
  GO TO 1
```

There is actually no need for the arrays X1 and X2 to be included in the subroutine argument list except that it may be required initially to put their contents to zero.

The use of arrays to hold the coefficients and the storage block variables is almost mandatory with non-recursive filters which are usually of high order. A non-recursive filter with its weighting sequence $(h_0, h_1, \ldots, h_{N-1})$ held in the N-element real array H can be implemented by the following code in which the array X holds the storage block variables.

```
1  INPUT XI
   Y = XI * H(1)
   DO 2 I = 1, N-1
   Y = Y + H(N - I + 1) * X(N - I + 1)
2  X(N-I+1) = X(N-I)
   X(1) = XI
   OUTPUT Y
   GO TO 1
```

A single loop can, as illustrated here, be used both to form the output number and to shift the number from each storage block into the next.

When floating point arithmetic is used, the accuracy given by an ordinary small computer seems to be adequate for ordinary filtering purposes provided that recursive filters are implemented in second order section form. The same is true when fixed-point arithmetic is used, provided that proper care is taken in the programming. The main difficulty with programming a filter in integer arithmetic is that if the numbers become too large, arithmetic overflow occurs and heavy distortion arises. On the other hand, if the numbers become too small so that the signals have only a few significant digits, accuracy is lost through round-off error and the filter, in effect, introduces noise into the signal.

In most assembly languages, a fixed-point binary number of, say, sixteen bits is usually regarded either as a binary fraction, with the binary point at the extreme left, or as a binary integer with the binary point at the extreme right. The location of the binary point is, of course, a conceptual matter concerned with the interpretation given to the binary computer words; it plays no actual part in the working of the computer. In programming digital filters with fixed-point arithmetic it is usually satisfactory to regard the binary point as being situated three places from the extreme left of the binary number. This particularly suits recursive second order sections where the coefficients never normally exceed a magnitude of two.

Overflow can be avoided, without undue loss of accuracy, by scaling the output of each second order section before it is applied to the following section. The inter-block scaling factors can be chosen so the gain from the input of the filter to the input of each section just reaches unity at some frequency. This ensures that the amplitudes of the signals handled by the different sections are of the same order of magnitude all the way through the filter. Subroutine FREQ can be used to evaluate the gain characteristic between the filter input and the

input of each section by calling it with a succession of values of N. The gain coefficient following each section is then chosen so that the gain from the filter input to the input of the next section just reaches unity at some frequency.

In most cases, if arithmetic overflow occurs the worst that happens is that the filter output is distorted. However, under some conditions. arithmetic overflow in a recursive second order section can result in oscillation of the filter. That is, once overflow occurs, the output from the filter does not die away even though its input is made zero. This condition is perhaps best prevented by adjusting the amplitude of the input sequence so that overflow does not occur.

Chapter 2

Recursive Filter Design

Many digital filter applications call for gain characteristics which ideally would be of the brickwall type. Recursive filter designs can be produced which closely approximate gain characteristics of this type. A particular degree of approximation to a required gain characteristic can usually be produced by a recursive filter of much lower order than woulu be required if a non-recursive filter were used. Recursive filters are thus usually more economical in terms of computer time and memory than non-recursive filters in producing similar gain characteristics.

Unfortunately, recursive filters have the disadvantage that their phase characteristics can be quite non-linear. While this does not matter in many circumstances, it can be a serious drawback where waveforms must be distorted as little as possible as, for example, in ECG studies where the precise shape of the waveform may be important. Nevertheless, with some recursive filters such as bandstop filters having narrow stopbands, it is possible for the non-linearity of the phase characteristic to be concentrated about one point on the frequency scale and to result in little waveform distortion.

There are two principal approaches to the design of recursive digital filters. One method which has achieved some popularity involves first producing a suitable analogue filter design and then transforming this to give a corresponding digital filter design. This method has obvious attractions to people who are expert in analogue filter design but otherwise it introduces unnecessary complication. The other approach to recursive filter design, which is the approach taken in this chapter, is to construct a suitable digital filter directly.

If a lowpass filter is required, the design formulae given here, or the subroutines of the appendix, will enable a Butterworth or a Chebyshev lowpass filter design to be produced. The Butterworth and Chebyshev filter characteristics both provide approximations to the

behaviour of an ideal lowpass filter. For use in the design of highpass, bandpass or bandstop filters the concept of a *frequency transformation* (Constantinides, 1970), is used. The use of a suitable frequency transformation enables a bandpass filter, for example, to be produced by the process of first designing a suitable lowpass filter and then converting it to a bandpass design. Subroutines are given in the appendix to enable frequency transformations to be effected and these, together with the routines for the design of lowpass filters, enable virtually any form of brickwall filter characteristics to be produced with comparative ease.

BUTTERWORTH FILTERS

It is not possible to make a filter which exactly produces the gain characteristic of an ideal lowpass filter. However, it is possible to approximate this ideal behaviour as closely as desired. One digital filter gain characteristic which approximates the ideal lowpass behaviour is the Butterworth characteristic (Rader and Gold, 1967). The gain characteristic of an Nth order Butterworth filter is given by the expression

$$|H(e^{j2\pi fT})| = 1/\left\{1 + (\tan \pi fT/\tan \pi f_c T)^{2N}\right\}^{\frac{1}{2}} . \qquad 2.1$$

For frequencies less than f_c, $\tan \pi fT/\tan \pi f_c T$ is fractional and the gain is approximately unity. For frequencies exceeding f_c, where $\tan \pi fT/\tan \pi f_c T$ becomes large, the gain is close to zero. The higher the value of N, the filter order, the better is the approximation to the ideal lowpass characteristic. This can be seen from *Figure 9(a)* which shows the gain characteristics of Butterworth filters of orders 3, 6 and 12, each filter having a cut-off frequency equal to one tenth of the sampling frequency. The gain of a Butterworth filter falls uniformly, as the frequency is increased from zero to one half of the clock frequency. At the cut-off frequency, f_c, the gain lies at precisely 3 dB below its value at zero frequency.

A Butterworth filter of Nth order has N poles which lie on a circle in the z-plane. The poles are given by the values of β_m which fall within the unit circle where the real and imaginary parts, U_m and V_m, of β_m are given by

$$\left.\begin{aligned} U_m &= (1 - \tan^2 \pi f_c T)/d \\ V_m &= 2 \tan \pi f_c T \sin(m\pi/N)/d \\ \text{where } d &= 1 - 2 \tan \pi f_c T \cos(m\pi/N) \\ &\quad + \tan^2 \pi f_c T \end{aligned}\right\} \quad m = 0, 1, \ldots, 2N-1 .$$

If N is even, $m\pi/N$ should be replaced by $\pi(2m + 1)/2N$ in these formulae. An Nth order Butterworth filter has N zeros which are all

situated at z = −1. *Figure 9(b)* shows the pole-zero patterns of the Butterworth filters whose gain characteristics are shown in *Figure 9(a).*

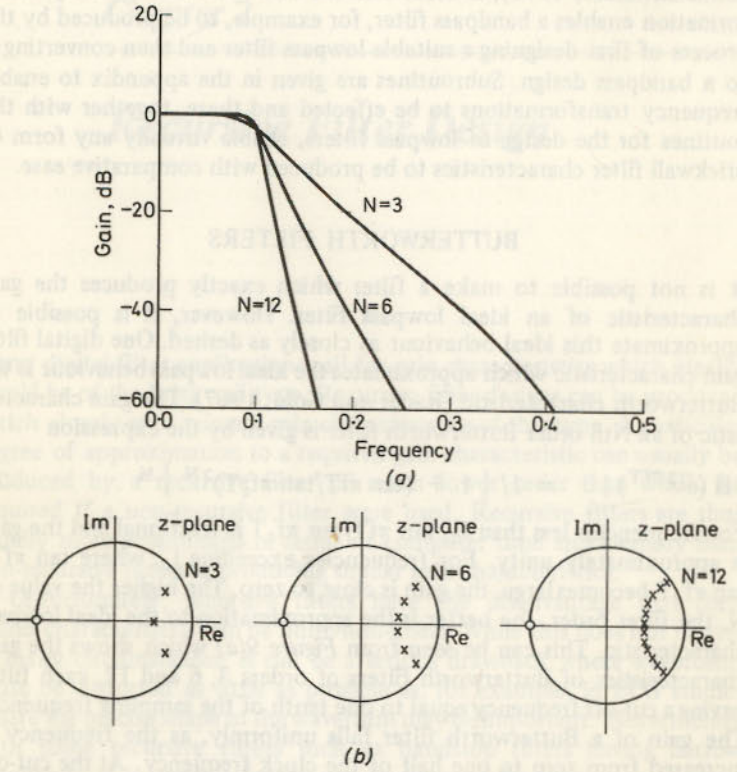


Figure 9. Characteristics of Butterworth lowpass filters of different orders. (a) *Gain characteristics.* (b) *Pole-zero plots*

The Fortran subroutine BUTTER which is given in the appendix computes the poles and zeros of a Butterworth filter of specified order N and cut-off frequency FC. After execution of the Fortran statement CALL BUTTER (N, FC, ALPHA, BETA), the complex arrays ALPHA and BETA contain the zeros and the poles of the required transfer function. The poles and zeros could be printed out if required. The arrays ALPHA and BETA can, after a call to subroutine BUTTER, be used directly as input to subroutine COEFF which will then yield as its output the coefficients of the second order transfer functions in a serial form realization of the Butterworth lowpass filter.

Example

A lowpass filter is required with the following specification :

Clock frequency = 1 kHz

Cut-off frequency = 50 Hz

Gain at zero frequency = 0 dB

Gain between 100 Hz and 500 Hz not to exceed −40 dB

First, the required filter order must be determined. This can be done by making use of equation 2.1 as follows. If the filter gain at 100 Hz does not exceed 0·01, or −40 dB on the decibel scale, then it certainly will not exceed this figure at higher frequencies since the gain of a Butterworth lowpass filter falls uniformly as the frequency increases from zero to one half the clock frequency. Putting these values into equation 2.1 yields the equation

$$0{\cdot}01 = 1/\left\{1 + (\tan \pi 100 \times 10^{-3} / \tan \pi 50 \times 10^{-3})^{2N}\right\}^{\frac{1}{2}}$$

The solution of this equation for N gives N = 6·409. N, being the order of a filter, must be an integer, so the lowest order of Butterworth filter which will give the required performance is seven. Alternatively, the required filter order N could have been found simply by evaluating the frequency response for successive values of N until eventually a satisfactory characteristic was obtained.

Subroutine BUTTER gives the poles of this filter as :

0·88987 + j0·28189	0·79742	+	j0·20257
0·88987 − j0·28189	0·79742	−	j0·20257
0·74393 + j0·10488			
0·74393 − j0·10488	0·72654	+	j0

Subroutine COEFF was used to compute the coefficients of a serial realization of the filter. These coefficients provided the input to subroutine FREQ which was used to compute the resulting gain and phase characteristics. *Figure 10* shows the computed gain characteristic of the filter. It can be seen that it does indeed meet the specification.

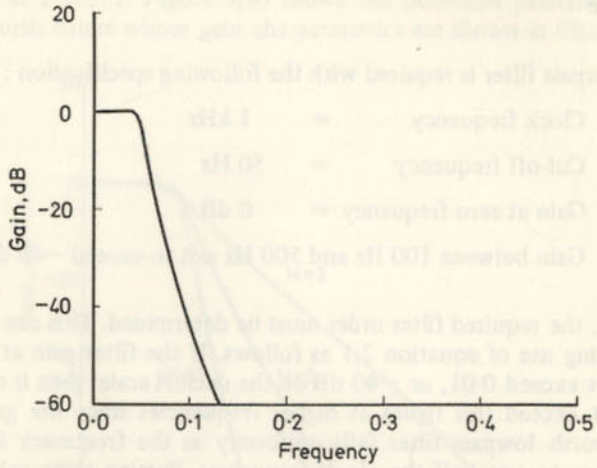


Figure 10. Gain characteristic of Butterworth example. Cut-off frequency 50 Hz, clock frequency 1 kHz, order 7

CHEBYSHEV FILTERS

An alternative to the Butterworth frequency response is the Chebyshev characteristic. The gain of a Chebyshev filter often decreases more rapidly in the stopband than the gain of a Butterworth filter of the same order. But the Chebyshev gain characteristic has ripple in the passband which may be considered a disadvantage in some applications.

The gain characteristic of a Chebyshev filter of order N is given by the expression

$$|H(e^{j2\pi fT})| = 1/\left\{1 + \epsilon^2 V_N^2 (\tan \pi fT/\tan \pi f_c T)\right\}^{\frac{1}{2}}$$

where ϵ is a design parameter and $V_N(x)$ is a Chebyshev polynomial of order N. The Chebyshev polynomial of order N can be found from the polynomials of order $N-1$ and $N-2$ by use of the recursion formula

$$V_N(x) = 2x\,V_{N-1}(x) - V_{N-2}(x).$$

The Chebyshev polynomial of any order can be found by starting with $V_1(x) = x$ and $V_0(x) = 1$. The amplitude δ of the ripple in the gain characteristic is given in terms of the design parameter ϵ by

$$\delta = 1 - 1/(1+\epsilon^2)^{\frac{1}{2}}.$$

If required, this equation can be solved to find the value of the design parameter ϵ corresponding to a given amplitude of passband ripple. The N poles of a Chebyshev lowpass filter of order N are given by the values of β_m which lie within the unit circle where U_m and V_m, the real and imaginary parts of β_m, are given by

$$U_m = 2(1 - a\tan \pi f_c T\cos\theta)/d - 1$$

$$V_m = 2\,b\tan \pi f_c T\sin\theta$$

where $d = (1 - a\tan \pi f_c T\cos\theta)^2 + b^2\tan^2 \pi f_c T\sin^2\theta$

$$a = \tfrac{1}{2}\left\{(\sqrt{1+1/\epsilon^2}+1/\epsilon)^{1/N} - (\sqrt{1+1/\epsilon^2}+1/\epsilon)^{-1/N}\right\}$$

$$b = \tfrac{1}{2}\left\{(\sqrt{1+1/\epsilon^2}+1/\epsilon)^{1/N} + (\sqrt{1+1/\epsilon^2}+1/\epsilon)^{-1/N}\right\}$$

$$\theta = m\pi/N,\ m = 0, 1, \ldots, 2N-1.$$

If N is even, $m\pi/N$ is to be replaced by $\pi(2m+1)/2N$ in these formulae. The N zeros all lie at $z = -1$, like those of a Butterworth lowpass filter.

Subroutine CHEBY evaluates these expressions to yield the poles and zeros of a Chebyshev lowpass filter. The input data to this subroutine are N, the order of the filter, f_c, the cut-off frequency as a fraction of the clock rate, and δ, the passband ripple. Subroutine CHEBY is used in a similar way to subroutine BUTTER; it yields as output the zeros and the poles of the Chebyshev filter held in the arrays ALPHA and BETA.

There is a trade-off involved in choosing the passband ripple δ; as δ is made larger, the attenuation in the stopband is increased, which is a good thing, but this is achieved at the expense of increased ripple amplitude in the passband gain characteristic which is normally undesirable. As a general rule, δ should be chosen as large as can be tolerated so as to get the greatest possible stopband attenuation.

Example

A Chebyshev filter is to be designed to the same specification as in the Butterworth design example with the additional condition that the passband gain should not vary by more than 1 dB, i.e. the ripple amplitude is to be 0·107. Subroutine CHEBY was used to compute the poles and zeros of filters with a number of different orders with the required cut-off frequency and passband ripple amplitude. Subroutine COEFF was then used to compute the second order section coefficients in each case and subroutine FREQ was used to compute the resulting gain and phase characteristics. It turned out that a fifth order Chebyshev

filter would achieve the specification, as can be seen from *Figure 11* which shows the gain characteristic. The poles of this filter, as computed by subroutine CHEBY, are

0·92582 + j0·29789

0·92582 − j0·29789

0·91136 + j0·17866

0·91136 − j0·17866

0·91183 + j0

There are also five zeros at z = −1. This example shows that by accepting passband ripple in the gain characteristic and by using a Chebyshev filter it is possible, in this case, to achieve the required attenuation characteristics with a filter of lower order than if a ripple-free gain characteristic is required. In this case a Chebyshev filter of

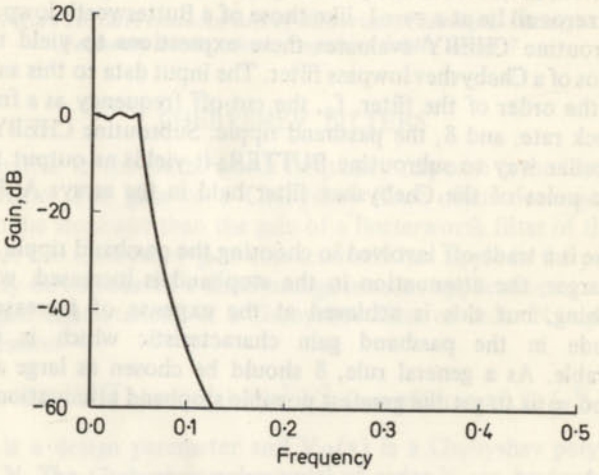


Figure 11. Gain characteristic of Chebyshev example. Cut-off frequency 50 Hz, clock frequency 1 kHz, order 5, passband ripple 1 dB

order five provides slightly better attenuation characteristics than the equivalent Butterworth filter of seventh order.

FREQUENCY TRANSFORMATIONS

A frequency transformation (Constantinides, 1970), enables a filter of one type to be transformed into one of some other type. For example,

a bandpass filter can be designed by first producing a suitable lowpass filter called the prototype. The application of a *lowpass-to-bandpass* frequency transformation to the lowpass prototype will then yield a bandpass filter design.

A very simple frequency transformation is the *lowpass-to-highpass.* This transformation is effected simply by taking a lowpass filter transfer function and replacing z by −z. If the original lowpass filter had a cut-off frequency f_c, the new highpass filter would then have a cut-off frequency at $1/2T - f_c$.

Example

A highpass filter is required with a cut-off frequency of 50 Hz. The clock rate is 1 kHz and it is thought that a second order Butterworth lowpass prototype will provide adequate performance.

The cut-off frequency of the highpass filter is to be 50 Hz, so the lowpass prototype must have a cut-off frequency f_c where $50 = 500 - f_c$; f_c is thus 450 Hz. Subroutine BUTTER was used to compute the poles and zeros of a second order Butterworth filter with this cut-off frequency and subroutine COEFF was used to compute its transfer function coefficients. The resulting transfer function is

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 + 1{\cdot}5610z^{-1} + 0{\cdot}6414z^{-2}}$$

The transfer function, G(z), of the required highpass filter, obtained by replacing z by −z in H(z) is thus given by

$$G(z) = \frac{1 - 2z^{-1} + z^{-2}}{1 - 1{\cdot}5610z^{-1} + 0{\cdot}6414z^{-2}}$$

Figure 12 shows the gain characteristic of the highpass filter as computed by subroutine FREQ.

This example illustrates the fact that the serial form block diagram of a highpass digital filter can be produced from the serial form block diagram of a lowpass filter simply by reversing the sign of the coefficients fed from the first block in each section. The application of the lowpass-to-highpass transformation leaves N, the order of the filter, unchanged.

For designing bandpass filters from lowpass prototypes, the lowpass-to-bandpass frequency transformation is used. If a bandpass filter with lower and upper cut-off frequencies f_1 and f_2 is required then a design can be produced by taking a lowpass filter whose cut-off frequency f_c

is $f_2 - f_1$ and replacing z^{-1} in its transfer function by the expression $-z^{-1}(z^{-1} - a)/(1 - az^{-1})$. The parameter a is given by the formula

$$a = \cos\{\pi(f_1 + f_2)T\}/\cos\{\pi(f_2 - f_1)T\}$$

In contrast to the lowpass-to-highpass transformation, the lowpass-to-bandpass transformation results in a filter whose order is double that of the prototype. For example, if the lowpass-to-bandpass transformation is applied directly to the second order sections of a lowpass prototype, fourth order sections result. To avoid problems of numerical accuracy,

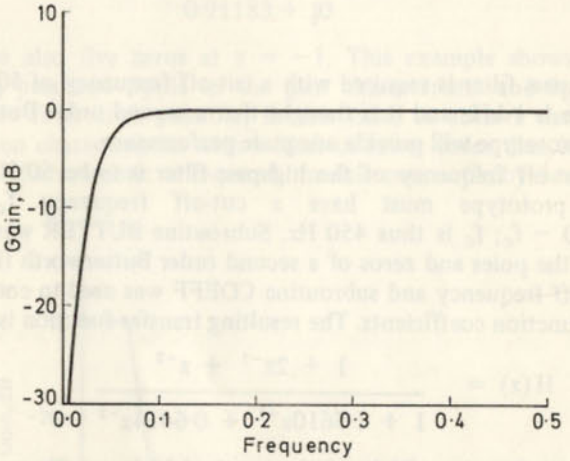


Figure 12. Gain characteristic of highpass example. Cut-off frequency 50 Hz, clock frequency 1 kHz, order 2

however, a final design consisting of second order sections is required. This difficulty can be resolved by applying the transformation to the poles and zeros of the lowpass prototype. For example, if the lowpass prototype has a zero at $z = \alpha$, then the bandpass filter must have zeros at values of z which satisfy the equation

$$-z^{-1}(z^{-1} - a)/(1 - az^{-1}) = \alpha^{-1}$$

This is a quadratic equation in z which can be solved by the usual quadratic equation solving formula. The solution shows that the bandpass filter must have zeros at

$$z = \tfrac{1}{2}(1 + \alpha)a \pm \{\tfrac{1}{4}(1 - \alpha)^2a^2 - \alpha\}^{\frac{1}{2}}. \quad 2.2$$

A similar formula gives the poles of the bandpass filter in terms of the poles of the lowpass prototype. Subroutine BPASS accepts as its input

the poles and zeros of a lowpass prototype in arrays as delivered by subroutines BUTTER or CHEBY. It computes the poles and zeros of the resulting bandpass filter by the use of expression 2.2. The bandpass filter can be realized as a cascade of second order sections by the use of the subroutine COEFF operating on the output of subroutine BPASS.

Example

A second order bandpass filter is required with a 3 dB bandwidth of 1 Hz; the lower and upper cut-off frequencies are to be 9·5 Hz and 10·5 Hz while the clock rate is 100 Hz.

Because the lowpass-to-bandpass transformation doubles the filter order, a first order lowpass prototype must be used. Its cut-off frequency must be $10{\cdot}5 - 9{\cdot}5 = 1$ Hz. Subroutine BUTTER gives the zero and the pole of this filter as −1·0, 0·93906 respectively.

Subroutine BPASS takes the zero and pole as input and yields the two zeros and the two poles of the resulting bandpass filter as

zeros	poles
1·0	0·78475 + j0·56853
−1·0	0·78475 − j0·56853

Subroutine COEFF yields the coefficient values for the second order section transfer function. The resulting transfer function is

$$\frac{1 - z^{-2}}{1 - 1{\cdot}5695z^{-1} + 0{\cdot}9391z^{-2}}$$

Figure 13 shows the gain characteristic as computed by subroutine FREQ. It is evident that the required shape of frequency response has been obtained. If a more rectangular gain characteristic had been required, a higher order lowpass prototype would have been used.

Another useful frequency transformation is the lowpass-to-bandstop transformation which enables bandstop filters to be designed from lowpass prototypes. A bandstop filter whose lower and upper cut-off frequencies are f_1 and f_2 can be produced by first designing a lowpass prototype filter whose cut-off frequency is $1/2T + f_1 - f_2$ and then replacing z^{-1} in its transfer function by $z^{-1}(z^{-1} - a)/(1 - az^{-1})$, where a is given by

$$a = \cos\{\pi(f_1 + f_2)T\}/\cos\{\pi(f_2 - f_1)T\}$$

The lowpass-to-bandstop transformation, like the lowpass-to-bandpass transformation, produces a filter whose order is twice that of the

prototype. To avoid the problem of implementing fourth order sections, this transformation can also be directly applied to the zeros and poles of the lowpass prototype. If the lowpass prototype has a zero at $z = \alpha$, the bandstop filter will have zeros at

$$z = \tfrac{1}{2}(1 - \alpha)a \pm \left\{ \tfrac{1}{4}(1 - \alpha)^2 a^2 + \alpha \right\}^{\frac{1}{2}}$$

and similarly with the poles.

The subroutine BSTOP can be used for designing bandstop filters. It accepts as input the N poles and N zeros of a lowpass prototype and yields the 2N poles and 2N zeros of the bandstop filter.

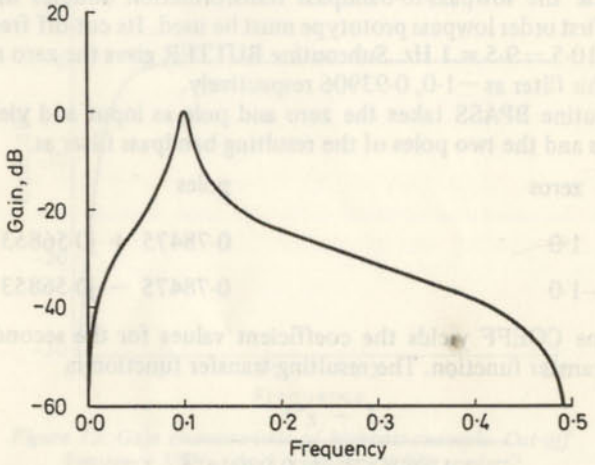


Figure 13. Gain characteristic of bandpass example. Cut-off frequencies 9·5 Hz and 10·5 Hz, clock frequency 100 Hz, first order Butterworth lowpass prototype

The 'centre frequency', f_0, of the bandstop filter lies between f_1 and f_2, and can be found by solving the equation

$$\cos 2\pi f_0 T = \cos\left\{ \pi(f_1 + f_2)T \right\} / \cos\left\{ \pi(f_2 - f_1)T \right\} . \qquad 2.3$$

It is sometimes important to be able to specify the centre frequency because at this frequency the bandstop filter has the same gain as the lowpass prototype has at 1/2T, one half the clock frequency. Any Butterworth or Chebyshev lowpass filter has zero gain at $f = 1/2T$ and so if one of these filter types is used as the lowpass prototype, the bandpass filter will have zero gain at f_0.

Example

A filter to reject mains hum is required. It should have zero gain at 50 Hz and the total width of the stopband should be 5 Hz. The clock frequency is to be 1 kHz.

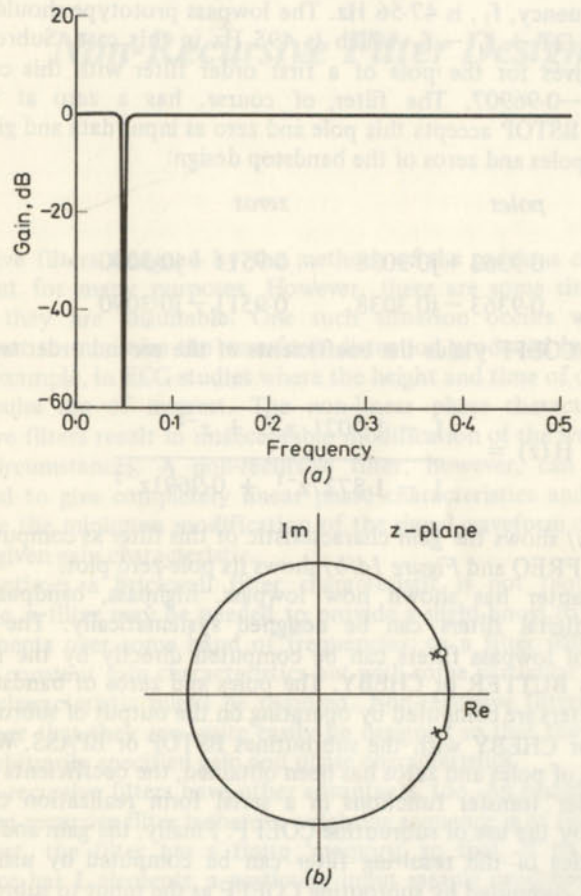


Figure 14. Bandstop example. Centre frequency 50 Hz, stopband width 5 Hz, clock frequency 1 kHz. (a) *Gain characteristic.* (b) *Pole-zero plot*

The stopband width or the difference between the upper and lower cut-off frequencies is to be 5 Hz. This gives $5 = f_2 - f_1$ or

$f_1 = f_2 - 5$. The centre frequency f_0 is to be 50 Hz and the clock period is 1 millisecond so substitution in equation 2.3 gives

$$\cos 2\pi\, 50 \times 10^{-3} = \cos \left\{ \pi(2f_2 - 5)10^{-3} \right\} / \cos \pi\, 5 \times 10^{-3}.$$

Solution of this equation gives $f_2 = 52{\cdot}56$ Hz and thus the lower cut-off frequency, f_1, is 47·56 Hz. The lowpass prototype should have cut-off at $1/2T + f_1 - f_2$ which is 495 Hz in this case. Subroutine BUTTER gives for the pole of a first order filter with this cut-off frequency −0·96907. The filter, of course, has a zero at −1·0. Subroutine BSTOP accepts this pole and zero as input data and gives as output the poles and zeros of the bandstop design:

poles	*zeros*
0·9363 + j0·3038	0·9511 + j0·3090
0·9363 − j0·3038	0·9511 − j0·3090

Subroutine COEFF yields the coefficients of the second order transfer function

$$H(z) = \frac{1 - 1{\cdot}9021\ z^{-1} + z^{-2}}{1 - 1{\cdot}8727z^{-1} + 0{\cdot}9691z^{-2}}$$

Figure 14(a) shows the gain characteristic of this filter as computed by subroutine FREQ and *Figure 14(b)* shows its pole-zero plot.

This chapter has shown how lowpass, highpass, bandpass and bandstop digital filters can be designed systematically. The poles and zeros of lowpass filters can be computed directly by the use of subroutines BUTTER or CHEBY. The poles and zeros of bandstop or bandpass filters are computed by operating on the output of subroutines BUTTER or CHEBY with the subroutines BSTOP or BPASS. When a suitable set of poles and zeros has been obtained, the coefficients of the second order transfer functions in a serial form realization can be computed by the use of subroutine COEFF. Finally, the gain and phase characteristics of the resulting filter can be computed by using the coefficients computed by subroutine COEFF as the input to subroutine FREQ.

Chapter 3

Non-Recursive Filter Design

Recursive filters designed by the methods of the previous chapter are excellent for many purposes. However, there are some situations in which they are unsuitable. One such situation occurs when it is important to minimize the waveform distortion produced by the filter as, for example, in ECG studies where the height and time of occurrence of a pulse are of interest. The non-linear phase characteristics of recursive filters result in unacceptable modification of the waveform in such circumstances. A non-recursive filter, however, can easily be designed to give completely linear phase characteristics and hence to produce the minimum modification of the signal waveform compatible with a given gain characteristic.

Sometimes a brickwall filter characteristic is not required. For example, a filter may be needed to provide a slight boost to frequency components over some band of frequencies; or a filter with approximately constant gain characteristics but with some particular non-linear phase characteristic might be required. Non-recursive filters have the advantage that they can quite easily be designed so that they approximate arbitrarily specified gain and phase characteristics.

Non-recursive filters have other advantages, too. An essential feature of a non-recursive filter is that its weighting sequence is of finite length. In effect, the filter has a finite 'memory' so that, if its weighting sequence has L elements, a particular input sample produces no effect at all on the output after L clock periods. As a result, the effect of a large spurious input such as a switching transient disappears completely after a known length of time.

Another advantage of non-recursive filters is that, because there is no feedback involved, there is no possibility whatsoever of instability occurring. While, under normal conditions, a properly designed recursive filter will be stable, it may become unstable if arithmetic

overflow occurs, unless precautions are taken, as explained in chapter 1. With a non-recursive filter, stability is guaranteed under all circumstances.

The principal disadvantage of non-recursive filters as compared with recursive designs is that to achieve a given specification they generally require more computer memory and more arithmetic operations per clock pulse. In many situations this disadvantage will be outweighed by their ability to yield linear phase characteristics or by their ease of design. In implementing non-recursive filters of order more than about thirty, the amount of computation required can be reduced by the use of the *fast Fourier transform*. Instead of being programmed directly in the way described in the section on programming digital filters (p. 19) the output sequence can be computed by operating on successive blocks of input data with a fast Fourier transform program. Each transformed block is multiplied by the required frequency response function and is then inverse transformed. The resulting blocks are then added together to yield the output sequence. Full details of these procedures are given by Gold and Rader (1969).

There are three types of design method available for the design of non-recursive filters to achieve a specified frequency response. The 'window method', which is presented in the following sections is simple and straightforward and will generally produce a satisfactory design. Its principal disadvantage is that, while the filter it produces will generally meet the given specification, it will not always be of the lowest possible order.

A second class of design methods is the use of analytical techniques analogous to the Butterworth and Chebyshev methods for designing recursive filters. Such design techniques (Herrmann, 1971), are actively being developed at the present time and may become widely used in future.

The third group of methods for the design of non-recursive filters depends on the use of a computer to produce a design by successive approximation techniques. These methods, which can produce very efficient designs, will doubtless become widely used as suitable computer programs become available (Parks and McClellan, 1972). A useful catalogue of lowpass and bandpass filter designs produced by this method is given by Rabiner, Gold and McGonegal (1970).

The frequency transformation techniques which were presented in the previous chapter are unfortunately not very useful in the design of non-recursive filters. If a frequency transformation is applied to a non-recursive prototype, a recursive filter design will generally result and the advantage of linear phase characteristics will be lost. An important exception here is the lowpass-to-highpass transformation which is applied simply by changing the signs of alternate elements of

the filter weighting sequence. The lowpass-to-highpass transformation can be applied directly to a lowpass non-recursive filter to produce a highpass non-recursive design. If the lowpass prototype filter has a linear-phase characteristic then the resulting highpass filter will be linear-phase too.

DESIGN BY THE WINDOW METHOD

The design method presented here, the window method (Gold and Rader, 1969), enables non-recursive filters to be designed to produce a frequency characteristic as close as required to an arbitrary specification. For the sake of simplicity, the method is presented only for the case of filters having an odd number of elements in their weighting sequences. This is not a great limitation for there can be very few situations in which, for example, it is much of a disadvantage to have to use a filter of order nineteen where one of order eighteen would do. In such a critical application it would, in any case, be advisable not to use the window method but instead to use a more efficient method of design in the hope of making a further reduction in the order of filter needed to meet the specification.

The window method starts from the ideal frequency response function to which an approximation is to be realized. The procedure consists of the following stages:

(1) The required frequency response function is used to compute, to a certain approximation, the ideal weighting sequence that the filter would have if there were no restriction on its length.

(2) The approximately computed ideal weighting sequence is modified by multiplying it by a 'window' function which reduces its length to some chosen value L.

(3) The frequency response function of the filter having this modified, or 'windowed', weighting sequence is evaluated. If the gain or phase characteristics do not present a close enough approximation to the desired response, the whole procedure is repeated with a more suitable window function.

Before discussing the process in detail, it is helpful to illustrate the steps that are involved with an example.

Figure 15(a) shows the ideal gain characteristic of a lowpass filter whose cut-off frequency is one quarter of the clock frequency. *Figure 15(b)* shows the ideal weighting sequence of a filter having this gain characteristic and zero phase shift at all frequencies. There are two points to note. First, the ideal weighting sequence is of infinite length so that this characteristic cannot be realized exactly by a non-recursive filter – nor, for that matter, by any filter at all. Secondly,

the weighting sequence exists for negative values of time and is symmetrical about t = 0; this is a consequence of zero phase shift having been specified at all frequencies.

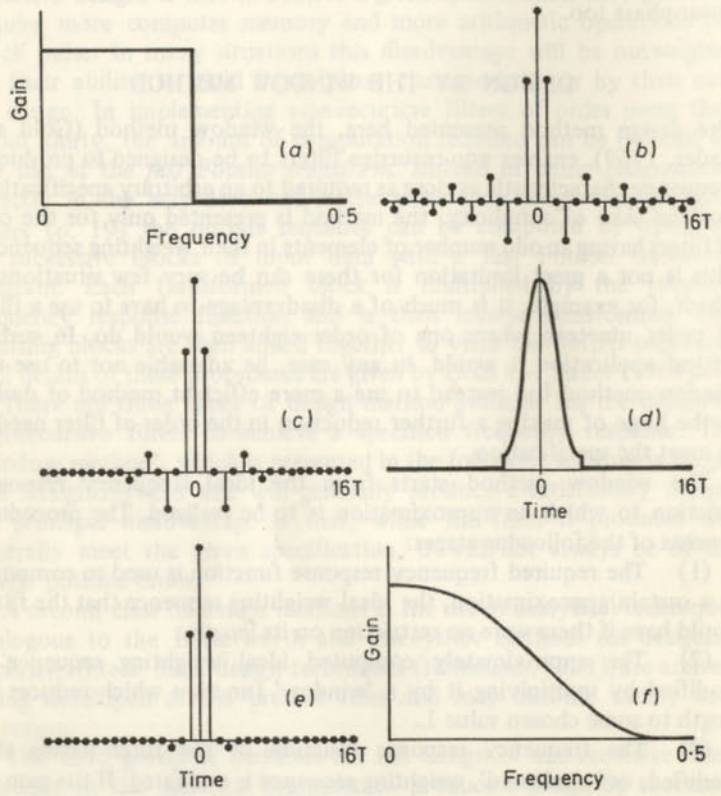


Figure 15. Non-recursive filter design by the window method: (a) *ideal gain characteristic;* (b) *ideal weighting sequence;* (c) *approximation to ideal weighting sequence computed by FFT;* (d) *Hamming window function, width L = 8T;* (e) *windowed weighting sequence;* (f) *gain characteristic of final design*

Figure 15(c) shows an approximation to the ideal weighting sequence computed by the discrete Fourier transform technique described in the following section. This approximate weighting sequence has been computed at 16 points. *Figure 15(d)* shows a *Hamming* window function and *Figure 15(e)* displays the result of multiplying the approximation to the ideal weighting sequence by the window function. The window is chosen to have a total length of 8 clock periods.

Finally, *Figure 15(f)* shows the gain characteristic corresponding to the modified weighting sequence. Whether or not this can be regarded as an adequate approximation to the ideal gain characteristic would entirely depend of course, on the requirements of the particular application. The weighting sequence which this design procedure has produced exists for negative values of time, as *Figure 15(e)* shows, and is therefore not physically realizable. However, the weighting sequence can be made physically realizable by the incorporation of a delay so that it is non-zero only for positive time values. The delay that needs to be incorporated for physical realizability does not affect the gain characteristic and only changes the phase characteristic by adding a phase shift proportional to frequency. The gain characteristic in *Figure 15(f)* is thus that of the final design. The individual stages in this design procedure are now discussed in more detail.

COMPUTING THE IDEAL WEIGHTING SEQUENCE

Subroutine FFT, which is given in the appendix, can be used to compute approximately the ideal weighting sequence from the required frequency response characteristic. This subroutine, in fact, computes the direct or the inverse discrete Fourier transform of an array of complex numbers. The discrete Fourier transform of an array of N complex numbers $(x_0, x_1, \ldots, x_{N-1})$, is another array of N complex numbers $(X_0, X_1, \ldots, X_{N-1})$, whose elements are given by the formula

$$X_n = \sum_{k=0}^{N-1} x_k e^{-j2\pi kn/N}, \quad n = 0, 1, \ldots, N-1. \qquad 3.1$$

The inverse of the discrete Fourier transform allows the array $(X_0, X_1, \ldots, X_{N-1})$, to be computed from $(X_0, X_1, \ldots, X_{N-1})$. The inverse is given by

$$x_k = 1/N \sum_{n=0}^{N-1} X_n e^{j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1. \qquad 3.2$$

Subroutine FFT evaluates these formulae using the fast Fourier transform (FFT) method which is much faster than direct evaluation of equation 3.1 or 3.2. The only slight drawback of the FFT method of computing discrete Fourier transforms is that N must be an integral power of two such as, for example, 16 or 512. This is no disadvantage in the present application as it does not preclude non-recursive filters from being designed with any length of weighting sequence whatsoever. Subroutine FFT is a development of the subroutine NLOGN presented by Robinson (1967).

To compute approximately the ideal weighting sequence, the required frequency response function is first evaluated at N points equally spaced from zero frequency to one point below the sampling frequency. The ideal filter characteristic shown in *Figure 15(a)* has zero phase shift and, depending on the frequency, either zero or unity gain. The evaluation of its response characteristic at 16 points spaced at one sixteenth of the clock frequency and starting at zero frequency gives the array (1,1,1,1,0·5,0,0,0,0,0,0,0,0·5,1,1,1). If it is necessary to evaluate the response characteristic at a point of discontinuity, the average of the values just to the left and to the right of the discontinuity is used. In this case, the average is 0·5. These values are placed in a Fortran complex array H. The imaginary part of each element of H is zero in this case. In most cases of nonrecursive filter design, zero phase shift characteristics will be required and so the numbers put into the array H will be purely real. If, however, a filter with non-zero phase shift characteristics is to be designed, such as, for example, a phase correcting filter, the following rules must be followed:

(1) The zero frequency value of the frequency characteristic (the contents of H(1)) must be purely real.

(2) The value of the frequency characteristic at 1/2T, half the sampling frequency (placed in H(N/2 + 1)) must also be purely real.

(3) The frequency characteristic at points between zero and half the sampling frequency, placed in (H(2), H(3), . . ., H(N/2)), can be specified as arbitrary complex numbers.

(4) The content of H(N/2 + 2) should be made the complex conjugate of the content of H(N/2); the number held in H(N/2 + 3) should be made the complex conjugate of H(N/2 − 1), and so on. The array H will thus have real-even, imaginary-odd symmetry about the element H(N/2 + 1).

These rules ensure that the resulting weighting sequence will be a sequence of purely real numbers.

With the present example, subroutine FFT is now used to compute the inverse discrete Fourier transform by use of the call statement CALL FFT(16, H, 1.0). After the call, the array H will contain the inverse discrete Fourier transform of its original contents. This will be an approximation to the ideal weighting sequence. The location H(1) contains the zpproximate value of the weighting sequence at t = 0, H(2) contains its approximate value at t = T, and so on to H(N/2 + 1), which will hold the approximate weighting sequence value at t = TN/2. The rest of the approximate weighting sequence is held in the second half of the array; H(N) contains the approximate weighting sequence value at t = −T, H(N − 1) contains its approximate value at t = 2T

and so on to H(N/2 + 1), which holds the approximate weighting sequence value at t = −TN/2. In the case of the present example the array which results is (0·5000, 0·3142, 0, −0·0935, 0, 0·0418, 0, −0·0124, 0, −0·0124, 0, 0·0418, 0, −0·0935, 0, 0·3142).

It is necessary to choose N, the number of points at which the required gain characteristic is specified. The choice of N is not critical. It must be an integral power of two in order to be compatible with the subroutine FFT and, at very least, must be equal to the window width, L, to be used. N is usually chosen so that the spacing between points on the frequency scale is a fraction such as one tenth or less, of the widths of the transition bands required in the final filter design. Such a value of N also provides a suitable number of points when the final frequency response is computed to enable it to be plotted easily as a smooth graph.

WINDOWING

The second stage in the design process involves the modification of the approximately computed ideal weighting sequence by multiplication with a window function. At this point the form and the width of the window function must be chosen.

Three shapes of window function are presented in this chapter: the rectangular, the Hamming and the Blackman windows. These window functions, of width L clock periods, are given by the following expressions:

(*a*) Rectangular window

$$w(t) = \begin{cases} 1, & |t| < LT/2 \\ 0, & |t| \geqslant LT/2 \end{cases}$$

(*b*) Hamming window

$$w(t) = \begin{cases} 0{\cdot}54 + 0{\cdot}46\cos(\pi t/LT), & |t| < LT/2 \\ 0, & |t| \geqslant LT/2 \end{cases}$$

(*c*) Blackman window

$$w(t) = \begin{cases} 0{\cdot}42 + 0{\cdot}5\cos(\pi t/LT) + 0{\cdot}08\cos(2\pi t/LT), & |t| < LT/2 \\ 0, & |t| \geqslant LT/2. \end{cases}$$

A window of width L clock periods is defined here as having the value zero at t = LT/2 and it thus results in a weighting sequence with L −1 non-zero elements. Subroutine WINDOW can be used to apply any of these window functions to a weighting sequence.

So far as brickwall type filters are concerned, the effect of modifying the ideal weighting sequence by multiplication with a window function has two effects. One effect is that the transition between passband and stopband is no longer abrupt. Instead, there is a transition band of finite width. The width of the transition band is inversely proportional to the width of the window, so that if a narrow transition band is required a long window must be used and, hence, a long weighting sequence will result. The width of the transition band also depends on the shape of window used, as can be seen from *Figure 16* which shows

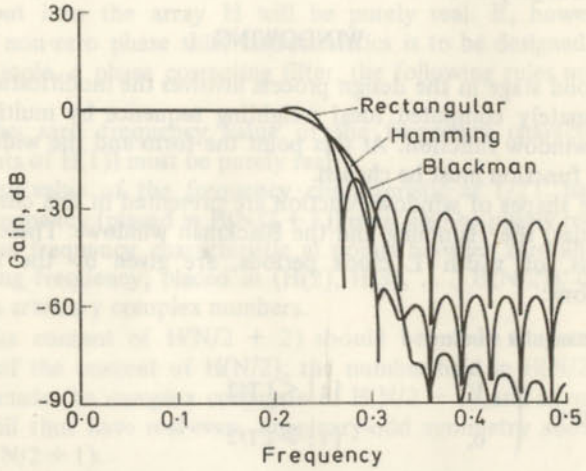


Figure 16. Gain characteristics of non-recursive filters designed using different window shapes. Window length 32 clock periods in each case

the gain characteristic of lowpass filters designed for cut-off at one quarter of the clock frequency, using rectangular, Hamming and Blackman windows of length 32 clock periods. The width of the transition band is approximately 2/LT, 4·5/LT or 6/LT, depending on whether the rectangular, Hamming or Blackman window is used.

The other effect of applying a window to the ideal weighting sequence is that the gain is no longer constant in the passband and is no longer

zero in the stopband. With the rectangular window the gain in the stopband is as large as 8 per cent of the passband gain. In other words, the stopband gain is only about 20 dB below the passband gain at one point in the stopband. As a result, the rectangular window is unsatisfactory for most purposes because of the poor stopband attenuation of filters designed with its use. With the Hamming window the stopband gain is more than 50 dB below the passband gain, while with the Blackman window, it is more than 70 dB down.

For the purpose of producing filter designs with narrow transition bandwidth for a given weighting sequence length, the rectangular window is best and the Blackman window is worst, with the Hamming window in between. On the other hand, for producing filters with good stopband rejection, the Blackman window is best and the rectangular window is worst while, again, the Hamming window has an intermediate performance. The appropriate window type is determined by the stopband gain that can be tolerated. For example, if the stopband gain were required to be at least 70 dB below the passband gain, the Blackman window would be used because the Hamming and rectangular windows do not provide the required stopband attenuation.

When the window type has been decided, it remains only to choose the window length L (measured in clock periods). The choice can be made from the foregoing relations between window type and length and stopband attenuation. For example, if a Hamming window were to be used, and the transition band were to be one twentieth of the clock frequency, then a window of length at least ninety clock periods would be needed since the transition bandwidth with the Hamming window in use is about four and one half times the reciprocal of the window length.

As mentioned before, subroutine WINDOW will apply a rectangular, Hamming or Blackman window to a sequence held in a complex array. Normally, of course, a weighting sequence will be real but it will be held in a complex array as a result of having been computed by subroutine FFT. In the case of the example, the call to WINDOW could be CALL WINDOW (16, 8, H, 0). The length of the input array is 16 and a window of length 8 is to be applied. The 0 specifies that a Hamming window is required. The resulting output array is (0·5000, 0·2725, 0, −0·0206, 0, 0, 0, 0, 0, 0, 0, 0, 0, −0·0206, 0, 0·2725). The positive-time part of the weighting sequence lies in the first half of the array while the negative-time part lies in the second half of the array. After the weighting sequence has been shifted to the right to make it physically realizable the resulting filter weighting sequence is (−0·0206, 0, 0·2725, 0·5000, 0·2725, 0, −0·0206).

EVALUATION OF RESULTING FILTER

When the windowed weighting sequence has been produced, it only remains to see whether the resulting gain and phase characteristics are satisfactory. This can be done by using subroutine FFT to operate directly on the output array delivered by subroutine WINDOW. The discrete Fourier transform of this array gives the frequency response function at N equally spaced frequencies from zero to (N −1)/NT − just below the clock frequency. The frequency response from zero to the Nyquist frequency (inclusive), is usually required. The first N/2 + 1 elements of the array hold the values of the frequency response function for this range of frequencies. If the ideal frequency response function was chosen to be purely real because zero phase shift was specified at all frequencies, the final frequency response function, though not ideal, should still be purely real. The call CALL FFT(16, H, −1.0), where H initially contains the output form the call to WINDOW described in the previous section, results in the array H containing (1·0037, 0·9877, 0·9144, 0·7466, 0·5000, 0·2534, 0·0856, 0·0124, −0·0037 0·0124, 0·0856, 0·2534, 0·5000, 0·7466, 0·9144, 0·9877).

If it is required to have the gain characteristic in decibels and the phase characteristic in degrees, subroutine GAINPH can be used. This subroutine accepts as input the values of the frequency response function held in the complex array H as delivered by subroutine FFT and it delivers as output the gain and phase at N/2 + 1 frequencies from zero to one-half the clock rate held in the real arrays GAIN and PHAS. For the present example, the statement CALL GAINPH (9, H, GAIN, PHAS) results in the array GAIN having the contents (0·03, −0·11, −0·78, −2·54, −6·02, −11·92, −21·35, −38·17, −48·65). The array PHAS contains, except for very small rounding errors, a zero in each location because the filter is a zero-phase filter and its frequency response function is real as a result.

If a brickwall type filter is required and the filter which results fails to provide a sufficiently good approximation to the ideal characteristics, it will be that the transition bands are too wide or that the stopband attenuation is insufficient. In the first case the window width should be increased. In the second case, a window shape giving better stopband attenuation should be used. If the same window width is used, this will result in the transition bands of the filter being made wider. This can be counteracted by the use of a larger window length so that the desired width of transition band is retained.

The discussion of non-recursive filter design in this chapter has

been in terms of brickwall type filter characteristics. However, the window method is equally applicable to the design of filters with more gently changing characteristics. For example, the next chapter describes how a filter to equalize the characteristics of a tape recorder can be designed using this method.

Chapter 4

Applications

Until now the number of cases in which digital filters have been applied to medical and biological problems has not been large. Part of the reason for this is that it is only recently that small computers have become widely used in medical laboratories. Also, it is only now that the possibilities of digital filtering techniques are becoming widely known. The purpose of this chapter is not so much to document existing applications but rather to suggest possible applications for digital filters that, for the most part, can be designed straightforwardly by the methods of the previous two chapters.

NOISE REDUCTION

Probably the most common application of filtering, either analogue or digital, is for the purpose of reducing or eliminating 'noise', that is, unwanted signals of any form which have become added to a wanted signal (Lynn, 1971). Noise can, of course, take many forms, from slowly changing voltages produced by electrode contact potentials, to the broad spectrum random noise produced in high gain amplifiers. In many cases, noise can be reduced by the use of a filter without too great an unwanted change in the signal itself. The change that the filter does effect on the signal, such as smoothing of any sharp edges, may be an acceptable price to pay for the resultant reduction in noise. The improvement in signal-to-noise ratio produced by a filter depends on having the signal and the noise occupy separate frequency ranges or at least on having the energies of the signal and the noise concentrated in different regions on the frequency scale. For example, if the signal occupies a narrow band of frequencies and the spectrum of the noise is essentially flat, like amplifier noise, then an appropriate bandpass filter can pass the signal almost unchanged while eliminating

much of the noise. At the other extreme, if the signal's spectrum covers a range of frequencies and the noise is predominantly 50 Hz mains hum then a 'notch' filter, i.e. a bandstop filter with a narrow stopband such as that described in chapter two, can be used to eliminate the hum while producing very little change in the signal. The essence of noise reduction by filtering is to accentuate those frequencies where the signal is strong relative to the noise and to attenuate those frequencies where the noise is strong relative to the signal.

It is worth remembering that there are sometimes methods other than filtering which will improve a signal-to-noise ratio. For example, if a signal of low amplitude is contaminated by occasional large narrow spikes of interference which greatly exceed the signal amplitude, then the noise can be greatly reduced by clipping at the peak signal level. This leaves the signal unaltered but reduces the heights of the interference spikes. Of course, the ideal way of avoiding noise is to prevent it ever entering the signal chain but, needless to say, this is not always possible.

Probably the most difficult problem in designing a filter to eliminate noise lies in choosing its specification. Once this has been done it is a relatively straightforward procedure to produce a working filter by using the methods of the previous chapters. The difficulty is that the optimum filter characteristic depends on the forms of the signal spectrum and the noise spectrum and on their relative levels. These data are usually unknown except in vague terms such as 'the noise spectrum is roughly flat and the signal contains no frequencies of interest above 10 Hz'. Fortunately, a filter specification which is chosen on the basis of common sense frequently proves to give nearly as great an improvement in signal-to-noise ratio as a mathematically optimum filter.

Sometimes extra constraints need to be introduced to ensure that a filter design is practicable. In the case of a filter to eliminate 50 Hz mains hum it might be argued that since the noise contains just one single frequency component at 50 Hz, a bandstop filter having an arbitrarily small stopband centred at 50 Hz could be used. For example, why not use a filter with a stopband having a width of 0·0001 Hz to eliminate the noise while, for all practical purposes, leaving any signal other than a pure tone at 50 Hz unaltered? There are at least two practical reasons why such a specification would not be used. First, the frequency of the hum or the centre frequency of the filter may vary from the nominal value of 50 Hz. The mains frequency may vary from its nominal figure or, if the signal comes from a tape recorder, hum recorded together with the signal will alter in frequency if the tape playback speed is not exactly the same as the recording speed. The

centre frequency of the filter stopband will change if the clock frequency varies from its nominal figure. With such a small stopband width as 0·0001 Hz, only a very small change would be needed in the mains frequency or the clock frequency to make the noise fall altogether outside the filter stopband. The second reason why an extremely narrow stopband would not be specified is that the more selective a digital filter is, the more sensitive it becomes to computational rounding error. In the case of a bandstop filter, the coefficients in the numerator of the transfer function become nearly equal to those in the denominator when the stopband is very narrow. Even with twelve or sixteen binary digit accuracy, the differences between the numerator and denominator coefficients may become lost as the result of rounding-off error if too narrow a stopband is specified.

In practice, a stopband width large enough to provide adequate attenuation for all likely variations of the mains and clock frequencies such as 1 Hz or 2 Hz would be used.

Problems produced by mains hum voltages can often be alleviated by an appropriate choice of sampling frequency. For example, if a sampling frequency of 50 Hz is used, mains hum at the same frequency will appear indistinguishable from a steady offset bias. This is due to the ambiguity ('aliasing'), involved in defining the frequency of a number sequence as discussed in the section on sinusoidal sequences and frequency response (page 19). It may well be possible to ignore this apparent offset in subsequent analysis. A sampling frequency of 60 Hz will result in a 50 Hz hum input appearing indistinguishable from a 10 Hz signal; 10 Hz is a frequency in the centre of the range of interest with many medical and biological signals and the aliased mains hum may obscure wanted signals.

SIGNAL ANALYSIS

Spectrum analysis is a very commonly used way of analysing signals. The general principle is to analyse the distribution of the energy of a signal with frequency. The fast Fourier transform is now widely used to effect spectrum analysis where fine frequency resolution is required. Frequently, however, fine frequency resolution is not required. It is then often more straightforward to use digital filters to perform the analysis rather than the fast Fourier transform.

Such an example is in EEG analysis, where it may be required to measure the activity in δ, θ, α and β bands. What is required is a set of bandpass filters each of which has a passband corresponding to the frequencies of one of these bands. The signal to be analysed is applied

to the inputs of the filters and the amplitude of the output of each filter provides a measure of the activity in each band.

The precise shape of the filter that should be used for EEG analysis and even the cutoff frequencies that should be used are matters on

TABLE 2

Filter	*Cutoff frequencies (Hz) (clock frequency=100 Hz)*	*Second-order section transfer function coefficients*		*Gain coefficients*
		b_1	b_2	
Delta	1·0 – 3·5	−1·931122	0·944569	$4{\cdot}8965 \times 10^{-7}$
		−1·988421	0·992294	
		−1·966323	0·972292	
		−1·924864	0·973091	
		−1·907514	0·938061	
Theta	3·5 – 7·5	−1·813728	0·912491	$4{\cdot}868 \times 10^{-6}$
		−1·934895	0·982203	
		−1·883164	0·944947	
		−1·747191	0·963006	
		−1·753766	0·913390	
Alpha	7·5 – 14·0	−1·482760	0·860451	$5{\cdot}0874 \times 10^{-5}$
		−1·754853	0·967726	
		−1·643895	0·905381	
		−1·235545	0·945516	
		−1·317558	0·869706	
Beta	14·0 – 22·0	−0·804375	0·829831	$1{\cdot}3721 \times 10^{-4}$
		−1·249232	0·953828	
		−1·071013	0·873283	
		−0·357596	0·941354	
		−0·528662	0·853527	

which there is no agreed standard. However, as an example of digital filters that can be used for EEG analysis, a set of bandpass filters was designed by use of the methods of chapter two and the subroutines given in the appendix. Fifth order Chebyshev lowpass prototypes were used with 0·5 dB passband gain ripple ($\delta = 0{\cdot}056$), and the clock frequency was assumed to be 100 Hz. The resulting bandpass filters were thus of tenth order with the same gain ripple. The gain

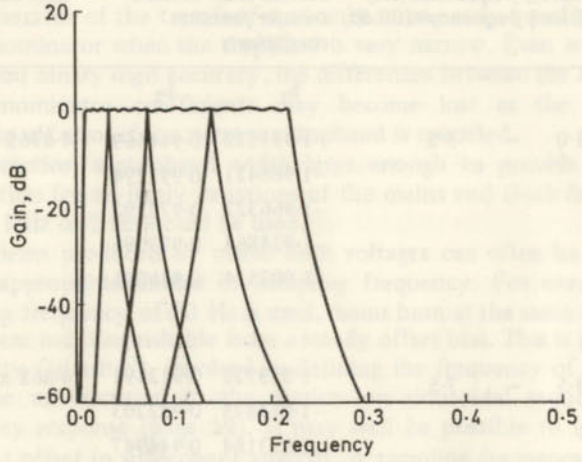


Figure 17. Gain characteristics of EEG band filters

characteristics of these filters are shown in *Figure 17.* Each filter transfer function is the product of a gain coefficient, a_0, and five second order section transfer functions of the form

$$H(z) = \frac{1 - z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

Table 2 gives the cutoff frequencies of each filter, the coefficients b_1 and b_2 in the second order section transfer functions, and the value of gain coefficient to make the maximum gain in the passband unity.

DECIMATION

When a signal is digitized, the minimum sampling frequency that can be used is determined by the effective bandwidth of the signal rather than by the highest frequency which happens to be of interest, as

explained in the section on Digitization (page 2). For example, an ECG signal contains components of high frequency because of its spiky character even though the heart rate, which may be the quantity of prime interest, is itself low. If too low a sampling frequency is used, the peaks of the ECG waveform may be altogether missed. If a signal is to be stored digitally it is an advantage to have the minimum possible sampling rate so as to minimize the number of digits that need to be stored for a given length of signal.

If only the low frequency components of a signal are of interest, the signal can be passed through a suitable analogue lowpass filter prior to sampling to limit the signal bandwidth, and then a low clock frequency can be used. This has the disadvantage that a suitable analogue lowpass filter must be available for each sampling frequency that may be needed. Alternatively, the signal can be digitized with a high sampling frequency and then digitally lowpass filtered to remove frequency components lying above the highest frequency of interest. The output of the digital lowpass filter can then be 'decimated', that is, its output can be resampled by preserving only every nth output number and discarding the rest. If the original sampling rate is 100 Hz and only every eighth number at the output of the digital lowpass filter is preserved, then an effective overall sampling rate of 12·5 Hz is produced. The digital lowpass filter would be designed to have a cutoff frequency at somewhere below one half the effective sampling rate; in this case its cutoff frequency could well be chosed to be 5 Hz, if it had a sharp cutoff characteristic.

An advantage of this decimation scheme is that if the effective overall sampling frequency is to be changed, the digital filter can easily be changed to suit. Another advantage is that a non-recursive filter with linear phase characteristics yet having a sharp cut-off can be used if it is required to minimize waveform distortion. Analogue filters having sharp cut-off characteristics inevitably produce significant waveform changes as a result of their non-linear phase characteristics.

EQUALIZATION

By the time a signal reaches the analogue-to-digital converter of the computer in which it is to be processed it will inevitably have undergone some modification as a result of, possibly, having been recorded and replayed on a magnetic tape recorder or as a result of the non-ideal characteristics of the transducer used to convert the physiological signal of interest to an electrical one. To some extent it is possible to use a digital filter to counteract such modifications of the signal.

It may be required, for example, to correct for the variation of gain with frequency of a tape recorder. This can be done by passing the digitized signal through a digital filter whose gain characteristic is the inverse of that of the tape recorder. In designing such a filter, the first stage is to measure the gain of the tape recorder at a number of equally spaced frequencies, from zero to one half the sampling frequency. This is easily done by recording sine waves of known amplitude and measuring their amplitude on playback. The number of frequencies at which the gain is measured should be sufficient so that the difference between the measured gain at any two adjacent points is smaller than the total gain variation which is acceptable. For example, if a final variation in gain of ± 0·1 dB over the working frequency range is required, the tape recorder gain characteristic should be measured at frequencies sufficiently closely spaced for there to be a difference of less than about 0·1 dB between any adjacent pair of points.

The next stage in the design process is to calculate the reciprocal of each of the measured gain values, the gain being measured as a ratio and not in decibels. This gives the required gain values of the digital equalization filter at each of the frequencies. A nonrecursive filter can now be produced by using the window method presented in chapter three. For the present application a rectangular window whose length is equal to the number of points on the required frequency can be used. Thus, the filter weighting sequence is computed by filling an array with the values of the required gain from zero to one half the sampling frequency and the values above this frequency (according to the rule given in the section which deals with computing the ideal weighting sequence: page 48); and then by computing the inverse discrete Fourier transform of the array. The result of this computation gives the right half of the weighting sequence held in the left half of the array and the left half of the weighting sequence held in the right half of the array, as already explained. If subroutine FFT is to be used to compute the inverse discrete Fourier transform the number of points on the required frequency response must be an integral power of two such as sixteen or sixty-four, for example.

Although a digital filter can, in principle, be used to equalize large variations in gain it is not recommended for this use. The reason for this is that while the gain, and hence the output signal amplitude, of a tape recorder usually fall at high frequencies, the noise level generally does not. An equalization filter which compensates for the gain of the tape recorder falling to low values at high frequencies will, itself, have correspondingly large values of gain there. Consequently, it will accentuate the noise at these frequencies and it will diminish the signal-to-noise ratio. In the foregoing description of equalization filter

design it was tacitly assumed that the cutoff frequency of the tape recorder lay above the Nyquist frequency and that the equalization filter had thus only to compensate for relatively small variations in gain which occur within the working frequency range of the recorder. If the Nyquist frequency lies above the cutoff frequency of the tape recorder it is desirable that the equalization filter should only compensate for variations in tape recorder gain up to its nominal cutoff frequency. Above this frequency, the values of the desired filter gain can be made equal to its value at the tape recorder cutoff frequency. The filter will then only equalize for variations in gain up to the tape recorder cutoff frequency, and above this frequency the gain of the overall recorder-filter combination will fall off in the usual way.

If the phase shift and gain characteristics of a device can be measured or are known then an equalizer can be designed to correct variations in phase shift as well as in gain. A nonrecursive filter to perform such a task can also be designed using the methods of chapter three. The only difference in designing a filter to correct phase variations as well as gain variations is that a complex desired frequency response function is used instead of a purely real one.

OTHER APPLICATIONS

The design methods presented in this book have dealt entirely with filters specified in the 'frequency domain'. These design methods produce filters whose frequency response characteristics approximate some desired form. For some applications, a frequency domain specification is inappropriate. Instead of designing a filter to have a specific frequency response one designs directly for it to have a specific weighting sequence.

There are applications where one wishes to detect automatically the occurrence of a particular waveshape in a signal. This can sometimes be done by the use of a *Matched filter,* 'matched' to a particular waveshape that is to be detected (Turin, 1960). A digital filter which matches a particular waveform or, strictly, which matches a particular number sequence, is one whose weighting sequence is a time-reversed version of the sequence it matches, apart from a possible change in amplitude scale and time origin. For example, suppose an indication is to be produced if a waveform whose sample values are (1·0, 1·5, –1·0, –2·0, 1·2), is applied to the input of an analogue-to-digital converter. A matched filter for this sample sequence could have the weighting sequence (12·0, –20·0, –10·0, –15·0, 10·0). The property which makes a matched filter useful for detecting signals of known waveshape is that it gives a larger output at some particular clock instant in

response to the sequence it matches than it does for any other sequence having the same duration and energy. By the 'energy' of a sequence is meant the sum of the squares of its elements. Thus, when the signal to which it is matched appears at the input to a matched filter, a large response is produced. Outputs exceeding some threshold can then be taken to indicate the presence of the waveform to be detected.

Since its weighting sequence is simply the time reverse of the sequence that is to be detected, the design of a matched filter is an almost trivial exercise. It can, in fact, be directly programmed as a nonrecursive filter.

In some applications, notably in EEG work, several signals may be processed at once. A theory of multi-channel digital filters (Robinson, 1967) has been developed, largely by geophysicists who need to process multiple signals which originate from an array of seismic sensors. A multi-channel filter is one which has several inputs or outputs. It seems likely that such multi-channel filters could be applied to the processing of multiple EEG signals. The use of a single multi-channel filter in such an application would, in principle, offer better performance than a number of single conventional digital filters used to process each EEG channel separately. However, this is perhaps more of an area where research is required than a situation in which the technique can immediately be put into use.

Another area of digital filter theory which is undergoing development at the present time is two-dimensional filtering (Shanks and Treitel, 1972). With two-dimensional filters the function to be filtered is no longer usually a time function but is, instead, a function of space, defined by x and y coordinates. Two-dimensional filtering can be applied to digitized images produced from x-ray films for purposes of removing noise or accentuating fine detail. There appears to be no reason why such filtering techniques should not be applied to processing two-dimensional fields measured on the body such as the distribution of EEG voltages over the head or the distribution of temperature over some region of the body.

The last few suggested applications would make use of 'unconventional' filter theory. However, there are undoubtedly numerous applications for conventional single-input/single-output digital filter techniques which remain to be put into practice.

References

Constantinides, A. G. (1970). 'Spectral Transformations for Digital Filters.' *Proc. Instn elect. Engrs* **117**, 1585

Gold, B. and Rader, C. M. (1969). *Digital Processing of Signals.* New York: McGraw-Hill.

Herrmann, O. (1971). 'On the Approximation Problem in Non-recursive Digital Filter Design.' *Instn elect. electron. Engrs, Trans. Circuit Theory*, **CT-18**, 411

Jackson, L. B. (1970). 'Roundoff-noise Analysis for Fixed Point Digital Filters Realised in Cascade or Parallel Form.' *Instn elect. electron. Engrs, Trans. Audio and Electroacoustics*, **AU-18**, 107

Lindorff, D. P. (1965). *Theory of Sampled-data Control Systems.* New York; Wiley.

Lynn, P. A. (1971). 'Recursive Digital Filters for Biological Signals.' *Med. Biol. Engng.*, **9**, 37

Parks, T. W. and McClellan, J. H. (1972). 'A Program for the Design of Linear Phase Finite Impulse Response Digital Filters.' *Instn elect. electron. Engrs, Trans. Audio and Electroacoustics*, **AU-20**, 115

Rabiner, L. R., Gold, B., McGonegal, C. A. (1970). 'An Approach to the Approximation Problem for Non-recursive Digital Filters.' *Instn elect. electron. Engrs, Trans. Audio and Electroacoustics*, **AU-18**, 83

Rader, C. M. and Gold, B. (1967). 'Digital Filter Design Techniques in the Frequency Domain.' *Proc. Instn elect. electron. Engrs*, **55**, 149

Robinson, E. A. (1967). *Multichannel Time Series Analysis.* San Francisco: Holden-Day

Shanks, J. L., Treitel, S. and Justice, J. H. (1972). 'Stability and Synthesis of Two-dimensional Recursive Filters.' *Instn elect. electron. Engrs, Trans. Audio and Electroacoustics*, **AU-20**, 195

Turin, G. L. (1960). 'An Introduction to Matched Filters.' *Instn radio. Engns Trans. on Information Theory*, **IT-6**, 311

Appendix

COMPUTER PROGRAMS

The Fortran subroutines given in this appendix are intended to enable the reader to design digital filters to suit his own applications. The subroutines fall into two groups; those for designing and evaluating recursive filters and those for dealing with non-recursive filters. The subroutines within each group are designed to be compatible in the sense that the output from one subroutine is in a form which is directly suitable as the input to another appropriate subroutine. The main program to design a digital filter thus needs consist of little more than the appropriate DIMENSION or COMPLEX statements, a READ statement to provide the appropriate input data, a succession of subroutine calls, and a suitable set of output commands.

The routines have been tested and used on an ICL 1904A computer. Very little modification, if any, should be required to make them suitable for any computer having a Fortran IV compiler. The reader who is fortunate enough to have the use of an interactive terminal will be able to use the programs for interactive filter design. The comment lines included in each subroutine should give sufficient information on the form of the input and output data. For ease of reference, the following table gives the names and functions performed by the subroutines.

RECURSIVE FILTER DESIGN ROUTINES

SUBROUTINE NAME	FUNCTION PERFORMED
BUTTER	Computes the poles and zeros of a Butterworth lowpass filter.
CHEBY	Computes the poles and zeros of a Chebyshev lowpass filter.

BPASS	Computes the poles and zeros of a bandpass filter from those of a lowpass filter.
BSTOP	Computes the poles and zeros of a bandstop filter from those of a lowpass filter.
COEFF	Computes the coefficients of the transfer functions of a serial form realization of a filter from its poles and zeros.
FREQ	Computes the gain and phase characteristics of a filter from the coefficients of the transfer functions of its serial form realization.

NON-RECURSIVE DESIGN ROUTINES

FFT	Computes the direct or inverse discrete Fourier transform of a complex array. It can be used to compute a weighting sequence from a frequency response and vice versa.
WINDOW	Applies a rectangular, Hamming or Blackman window to a complex array.
GAINPH	Computes the gain and phase characteristics from the frequency response function as computed by subroutine FFT.

```
      SUBROUTINE BUTTER (N,FC,ALPHA,BETA)
C
C     SUBROUTINE COMPUTES THE POLES AND ZEROS OF A
C     BUTTERWORTH LOWPASS DIGITAL FILTER.
C
C     INPUTS ARE:          N = ORDER OF FILTER REQUIRED.
C                         FC = REQUIRED CUTOFF FREQUENCY
C                              AS A FRACTION OF THE CLOCK
C                              FREQUENCY.
C
C     OUTPUTS ARE:  ALPHA = COMPLEX ARRAY CONTAINING
C                           THE TRANSFER FUNCTION ZEROS
C                           IN ITS FIRST N LOCATIONS.
C                          (ALL N ZEROS LIE AT Z=-1.0)
C                   BETA = COMPLEX ARRAY CONTAINING
C                           THE TRANSFER FUNCTION POLES
C                           IN ITS FIRST N LOCATIONS.
C                           COMPLEX POLES OCCUPY
C                           ADJACENT LOCATIONS; IF N IS
C                           ODD THE REAL POLE IS IN
C                           LOCATION 1.
C
      COMPLEX ALPHA(30),BETA(30)
      WC=3.141592654*FC
      TAN2=2.0*SIN(WC)/COS(WC)
      TANSQ=0.25*TAN2**2
      IF (N.EQ.1) GO TO 2
      IN=MOD(N,2)
      N1=N+IN
      N2=(3*N+IN)/2-1
      DO 1 M=N1,N2
      A=3.141592654*FLOAT(2*M+1-IN)/FLOAT(2*N)
      ANUM=1.0-TAN2*COS(A)+TANSQ
      U=(1.0-TANSQ)/ANUM
      V=TAN2*SIN(A)/ANUM
      I=(N2-M)*2+1
      BETA(I+IN)=CMPLX(U,V)
    1 BETA(I+IN+1)=CMPLX(U,-V)
      IF (IN) 3,3,2
    2 BETA(1)=CMPLX(((1.0-TANSQ)/(1.0+TAN2+TANSQ)),0.0)
    3 DO 4 I=1,N
    4 ALPHA(I)=(-1.0,0.0)
      N1=N+1
      DO 5 I=N1,30
      ALPHA(I)=(0.0,0.0)
    5 BETA(I)=(0.0,0.0)
      RETURN
      END
```



```
      SUBROUTINE CHEBY (N,FC,DELTA,ALPHA,BETA)
C
C     SUBROUTINE COMPUTES THE POLES AND ZEROS OF A
C     CHEBYSHEV LOWPASS DIGITAL FILTER.
C
C     INPUTS ARE:          N = ORDER OF FILTER REQUIRED.
C                         FC = REQUIRED CUTOFF FREQUENCY
C                              AS A FRACTION OF THE CLOCK
C                              FREQUENCY.
C                      DELTA = REQUIRED PASSBAND GAIN
C                              RIPPLE AMPLITUDE.
C
C     OUTPUTS ARE:  ALPHA = COMPLEX ARRAY CONTAINING
C                           THE TRANSFER FUNCTION ZEROS
C                           IN ITS FIRST N LOCATIONS.
C                          (ALL N ZEROS LIE AT Z=-1.0)
C                   BETA = COMPLEX ARRAY CONTAINING
C                           THE TRANSFER FUNCTION POLES
C                           IN ITS FIRST N LOCATIONS.
C                           COMPLEX POLES OCCUPY
C                           ADJACENT LOCATIONS; IF N IS
C                           ODD THE REAL POLE IS IN
C                           LOCATION 1.
C
      COMPLEX ALPHA(30),BETA(30)
      E=1.0/SQRT((1.0/((1.0-DELTA)**2)-1.0)
      X=(SQRT(E*E+1.0)+E)**(1.0/FLOAT(N))
      XI=1.0/X
      A=0.5*(X-XI)
      B=0.5*(X+XI)
      WC=3.141592654*FC
      TANA=SIN(WC)/COS(WC)
      TAN2=TANA**2
      IN=MOD(N,2)
      N1=N+IN
      N2=(3*N+IN)/2-1
      DO 1 M=N1,N2
      TH=3.141592654*FLOAT(2*M+1-IN)/FLOAT(2*N)
      ATC=A*TANA*COS(TH)
      BTS=B*TANA*SIN(TH)
      DEN=(1.0-ATC)**2+BTS**2
      U=2.0*(1.0-ATC)/DEN-1.0
      V=2.0*BTS/DEN
      I=(N2-M)*2+1
      BETA(I+IN)=CMPLX(U,V)
    1 BETA(I+IN+1)=CMPLX(U,-V)
      IF (IN) 3,3,2
    2 U=2.0/(1.0+A*TANA)-1.0
      BETA(1)=CMPLX(U,0.0)
    3 CONTINUE
      DO 4 I=1,N
    4 ALPHA(I)=(-1.0,0.0)
      N1=N+1
      DO 5 I=N1,30
      ALPHA(I)=(0.0,0.0)
    5 BETA(I)=(0.0,0.0)
      RETURN
      END
```

```
      SUBROUTINE BPASS (N,F1,F2,ALPHA,BETA)
C
C     SUBROUTINE COMPUTES THE 2N POLES AND ZEROS OF A
C     BANDPASS DIGITAL FILTER FROM THE N POLES AND
C     ZEROS OF A LOWPASS PROTOTYPE.
C
C     INPUTS ARE:        N = ORDER OF LOWPASS PROTOTYPE.
C                       F1 = LOWER CUTOFF FREQUENCY.
C                       F2 = UPPER CUTOFF FREQUENCY.
C                    ALPHA = ARRAY CONTAINING ZEROS OF
C                            LOWPASS PROTOTYPE.
C                     BETA = ARRAY CONTAINING POLES OF
C                            LOWPASS PROTOTYPE.
C     (THE LOWPASS PROTOTYPE SHOULD HAVE CUTOFF AT
C     FC = F2-F1. ALL FREQUENCIES EXPRESSED AS A
C     FRACTION OF THE CLOCK RATE.)
C
C     OUTPUTS ARE:  ALPHA = ARRAY CONTAINING THE 2N
C                           ZEROS OF THE BANDPASS FILTER.
C                    BETA = ARRAY CONTAINING ITS POLES.
C
      COMPLEX ALPHA(30),BETA(30),Z,S
      P=3.14159265359
      A=COS(P*(F1+F2))/COS(P*(F2-F1))
      IM=MOD(N,2)
      N1=N-IM
      N2=N
    1 DO 2 IA=1,2
      S=CMPLX((FLOAT(2*IA-3)),0.0)
      IB=2-IA
      DO 2 I=1,N1
      J=N2-I+1
      K=N2*(1+IB)-I+1+IM*(IA-1)
      Z=(0.5,0.0)*((1.0,0.0)+ALPHA(J))*CMPLX(A,0.0)
      ALPHA(K)=Z+S*CSQRT(Z*Z-ALPHA(J))
      Z=(0.5,0.0)*((1.0,0.0)+BETA(J))*CMPLX(A,0.0)
    2 BETA(K)=Z+S*CSQRT(Z*Z-BETA(J))
      IF (N1.EQ.N2) RETURN
      N1=1
      N2=1
      IM=0
      GO TO 1
      END
```



```
      SUBROUTINE BSTOP (N,F1,F2,ALPHA,BETA)
C
C     SUBROUTINE COMPUTES THE 2N POLES AND ZEROS OF A
C     BANDSTOP DIGITAL FILTER FROM THE N POLES AND ZEROS
C     OF A LOWPASS PROTOTYPE.
C
C     INPUTS ARE:        N = ORDER OF LOWPASS PROTOTYPE.
C                       F1 = LOWER CUTOFF FREQUENCY.
C                       F2 = UPPER CUTOFF FREQUENCY.
C                    ALPHA = ARRAY CONTAINING THE ZEROS
C                            OF THE LOWPASS PROTOTYPE.
C                     BETA = ARRAY CONTAINING THE POLES
C                            OF THE LOWPASS PROTOTYPE.
C     (THE LOWPASS PROTOTYPE SHOULD HAVE CUTOFF AT
C     FC = 0.5-F2+F1. ALL FREQUENCIES EXPRESSED AS A
C     FRACTION OF THE CLOCK RATE.)
C
C     OUTPUTS ARE:  ALPHA = ARRAY CONTAINING THE 2N
C                           ZEROS OF THE BANDSTOP FILTER.
C                    BETA = ARRAY CONTAINING ITS 2N
C                           POLES.
C
      COMPLEX ALPHA(30),BETA(30),Z,S,S1
      P=3.14159265359
      A=COS(P*(F1+F2))/COS(P*(F2-F1))
      IM=MOD(N,2)
      N1=N-IM
      N2=N
    1 DO 2 IA=1,2
      S=CMPLX((FLOAT(2*IA-3)),0.0)
      IB=2-IA
      DO 2 I=1,N1
      IN=MOD(I,2)*2-1
      S1=CMPLX(FLOAT(IN),0.0)
      J=N2-I+1
      K=N2*(1+IB)-I+1+IM*(IA-1)
      Z=(0.5,0.0)*((1.0,0.0)-ALPHA(J))*CMPLX(A,0.0)
      ALPHA(K)=S1*S*CSQRT(Z*Z+ALPHA(J))+Z
      Z=(0.5,0.0)*((1.0,0.0)-BETA(J))*CMPLX(A,0.0)
    2 BETA(K)=S*CSQRT(Z*Z+BETA(J))+Z
      IF (N1.EQ.N2) RETURN
      N1=1
      N2=1
      IM=0
      GO TO 1
      END
```

```
      SUBROUTINE COEFF (N,ALPHA,BETA,A1,A2,B1,B2,A0)
C
C     SUBROUTINE COMPUTES THE COEFFICIENTS IN A SERIAL-
C     FORM REALISATION OF A DIGITAL FILTER.
C
C                                  -1        -2
C                    N      1 + A1(I)Z  + A2(I)Z
C     H(Z) = A0 PRODUCT  ----------------------
C                I = 1              -1        -2
C                      1 + B1(I)Z  + B2(I)Z
C
C
C     INPUTS ARE:        N = NUMBER OF SECTIONS IN FILTER.
C                    ALPHA = ARRAY HOLDING FILTER ZEROS.
C                     BETA = ARRAY HOLDING FILTER POLES.
C     (CONJUGATE PAIRS MUST BE LOADED IN ADJACENT
C     LOCATIONS I,I+1 WHERE I IS AN ODD NUMBER).
C
C     OUTPUTS ARE: A1,A2,B1,B2: ARRAYS HOLDING SECTION
C                                          COEFFICIENTS.
C                      A0 = GAIN COEFFICIENT FOR UNITY
C                           GAIN AT ZERO FREQUENCY.
C     (IF FILTER HAS A ZERO AT Z=1, A0=1.0).
C
      COMPLEX ALPHA(30),BETA(30)
      DIMENSION A1(15), A2(15), B1(15), B2(15)
      A0=1.0
      DO 1 I=1,N
      I1=2*I-1
      I2=2*I
      A1(I)=REAL(-ALPHA(I1)-ALPHA(I2))
      A2(I)=REAL(ALPHA(I1)*ALPHA(I2))
      B1(I)=REAL(-BETA(I1)-BETA(I2))
      B2(I)=REAL(BETA(I1)*BETA(I2))
    1 A0=A0*(1.0+A1(I)+A2(I))/(1.0+B1(I)+B2(I))
      IF (ABS(A0).LT.1.0E-6) A0=1.0
      A0=1.0/A0
      RETURN
      END
```



```
      SUBROUTINE FREQ (N,NF,A1,A2,B1,B2,A0,GAIN,PHAS)
C
C     SUBROUTINE COMPUTES THE GAIN AND PHASE CHARACTER-
C     ISTICS OF A DIGITAL FILTER FROM THE COEFFICIENTS
C     OF ITS SECOND ORDER SECTION REALISATION.
C
C     INPUTS ARE:          N = NUMBER OF SECOND ORDER
C                              SECTIONS IN FILTER.
C                         NF = NUMBER OF POINTS REQUIRED
C                              ON FREQUENCY SCALE.
C                         A0 = GAIN COEFFICIENT.
C                 A1,A2,B1,B2: ARRAYS HOLDING SECOND ORDER
C                              SECTION COEFFICIENTS.
C
C     OUTPUTS ARE:      GAIN = ARRAY HOLDING VALUES OF
C                              FILTER GAIN IN DECIBELS.
C                      PHAS = ARRAY HOLDING VALUES OF
C                              FILTER PHASE SHIFT IN DEGREES.
C
C     THE GAIN AND PHASE ARE COMPUTED AT NF POINTS
C     EQUALLY SPACED FROM ZERO TO HALF THE CLOCK RATE
C     INCLUSIVE.
C
      COMPLEX WJ,WJ2,H,AD,AN
      DIMENSION A1(N), A2(N), B1(N), B2(N), GAIN(NF), PHAS(NF)
      C=180.0/3.141592654
      DO 3 J=1,NF
      OMEGA=3.141592654*FLOAT(J)/FLOAT(NF)
      WJ=CMPLX(COS(OMEGA),SIN(OMEGA))
      WJ2=WJ*WJ
      H=CMPLX(A0,0.0)
      DO 3 I=1,N
      AN=(1.0,0.0)+CMPLX(A1(I),0.0)*WJ+CMPLX(A2(I),0.0)*WJ2
      AD=(1.0,0.0)+CMPLX(B1(I),0.0)*WJ+CMPLX(B2(I),0.0)*WJ2
      X=CABS(AN)
      IF (X-1.0E-6) 4,4,1
    1 X=CABS(AD)
      IF (X-1.0E-6) 5,5,2
    2 H=H*AN/AD
      GAIN(J)=20.0*ALOG10(CABS(H))
      PHAS(J)=C*ATAN2(AIMAG(H),REAL(H))
    3 CONTINUE
      RETURN
    4 GAIN(J)=-120.0
      PHAS(J)=0.0
      GO TO 3
    5 GAIN(J)=120.0
      PHAS(J)=0.0
      GO TO 3
      END
```

```
      SUBROUTINE FFT (N,H,DIR)
C
C     SUBROUTINE COMPUTES THE DIRECT OR INVERSE DISCRETE
C     FOURIER TRANSFORM BY THE FFT METHOD.
C
C     INPUTS ARE:       N = LENGTH OF INPUT/OUTPUT ARRAY.
C                       H = COMPLEX ARRAY HOLDING
C                           SEQUENCE TO BE TRANSFORMED.
C                     DIR = -1.0 FOR DIRECT TRANSFORM
C                         =  1.0 FOR INVERSE.
C
C     OUTPUT IS IN ARRAY H AND REPLACES INPUT.
C
      DIMENSION M(20)
      COMPLEX H(N),WK,A,B
      VA=DIR*6.2831853070/FLOAT(N)
      LOG=1
      K=N
    1 K=K/2
      M(LOG)=K
      IF (K.EQ.1) GO TO 2
      LOG=LOG+1
      GO TO 1
    2 K=0
      DO 5 L=1,LOG
      NB=2**(L-1)
      LB=N/NB
      LBH=LB/2
      K=0
      DO 5 IB=1,NB
      V=VA*FLOAT(K)
      WK=CMPLX(COS(V),SIN(V))
      IS=LB*(IB-1)
      DO 3 I=1,LBH
      J=IS+I
      JH=J+LBH
      A=H(J)
      B=H(JH)*WK
      H(JH)=A-B
    3 H(J)=A+B
      DO 4 I=2,LOG
      IF (K.LT.M(I)) GO TO 5
    4 K=K-M(I)
    5 K=K+M(I)
      K=0
      DO 8 J=1,N
      IF (K.LT.J) GO TO 6
      A=H(J)
      H(J)=H(K+1)
      H(K+1)=A
    6 DO 7 I=1,LOG
      IF (K.LT.M(I)) GO TO 8
    7 K=K-M(I)
    8 K=K+M(I)
      IF (DIR.LT.0.0) RETURN
      A=CMPLX((1.0/FLOAT(N)),0.0)
      DO 9 I=1,N
    9 H(I)=H(I)*A
      RETURN
      END
```



```
      SUBROUTINE WINDOW (N,L,H,IWIND)
C
C     SUBROUTINE APPLIES A WINDOW OF TOTAL WIDTH L TO
C     A COMPLEX ARRAY.
C
C     INPUTS ARE:       N = LENGTH OF INPUT/OUTPUT
C                           ARRAY.
C                   IWIND = -1 FOR BLACKMAN WINDOW
C                         =  0 FOR HAMMING WINDOW
C                         =  1 FOR RECTANGULAR WINDOW.
C                       L = LENGTH OF WINDOW REQUIRED.
C                       H = COMPLEX ARRAY CONTAINING
C                           DATA TO BE WINDOWED.
C
C     OUTPUT REPLACES INPUT IN ARRAY H.
C
      COMPLEX H(N)
      IF (IWIND) 1,1,6
    1 DO 5 I=2,L/2
      A=6.28318531*FLOAT(I-1)/FLOAT(L)
      IF (IWIND) 3,2,2
    2 W=0.54+0.46*COS(A)
      GO TO 4
    3 W=0.42+0.5*COS(A)+0.08*COS(2.0*A)
    4 H(I)=H(I)*CMPLX(W,0.0)
      J=N-I+2
    5 H(J)=H(J)*CMPLX(W,0.0)
    6 DO 7 I=L/2+1,N-L/2+1
    7 H(I)=(0.0,0.0)
      RETURN
      END
```

```
      SUBROUTINE GAINPH (N,H,GAIN,PHAS)
C
C     SUBROUTINE COMPUTES THE GAIN AND PHASE
C     CHARACTERISTICS OF A NONRECURSIVE FILTER
C     FROM THE VALUES OF ITS FREQUENCY RESPONSE
C     INPUTS ARE:       N = NUMBER OF POINTS ON
C                           FREQUENCY RESPONSE.
C                       H = COMPLEX ARRAY HOLDING VALUES
C                           OF FREQUENCY RESPONSE
C                           FUNCTION.
C
C     OUTPUTS ARE:   GAIN = ARRAY CONTAINING GAIN IN DB.
C                    PHAS = ARRAY CONTAINING PHAS IN
C                           DEGREES.
C
C
      COMPLEX H(N)
      DIMENSION GAIN(N), PHAS(N)
      C=180.0/3.141592654
      DO 1 I=1,N
      X=CABS(H(I))
      IF (X.LT.1.0E-6) GO TO 2
      GAIN(I)=20.0*ALOG10(X)
      PHAS(I)=C*ATAN2(AIMAG(H(I)),REAL(H(I)))
    1 CONTINUE
      RETURN
    2 GAIN(I)=-120.0
      PHAS(I)=0.0
      GO TO 1
      END
```


Bibliography

General theory of digital filters

Hurewicz, W. (1947). 'Filters and Servo Systems with Pulsed Data.' In *Theory of Servomechanisms*, ed by H. M. James, N. B. Nichols and R. S. Phillips. New York; McGraw-Hill

Kaiser, J. F. (1966). 'Digital filters.' In *System Analysis by Digital Computer*, ed by F. F. Kuo and J. F. Kaiser, New York; Wiley

Linvill, W. K. (1951). 'Sampled-data Control Systems Studied Through Comparison with Amplitude Modulation.' *Trans. Am. Instn elect. Engrs*, **70**, Pt. 2, 1779

Ragazzini, J. R., and Franklin, G. F. (1958). *Sampled Data Control Systems*, New York; McGraw-Hill

Robinson, E. A. and Treitel, S. (1964). 'Principles of Digital Filtering.' *Geophysics*, **29**, 395

Special issues on digital filtering

Instn elect. electron. Engrs. Trans. Audio Electroacoustics, September, 1968, **AU-16**, 301 and June, 1970, **AU-18**, 81

Effects of round-off error

The following articles treat the problems which arise from finite word length in digital filters. The subject is rather complicated.

Gold, B. and Rader, C. M. (1967). 'Effects of Parameter Quantisation on the Poles of a Digital Filter.' *Proc. Instn elect. electron. Engrs*, **55**, 688

Knowles, J. B. and Olcayto, E. M. (1968). 'Coefficient Accuracy and Digital Filter Response.' *Instn elect. electron. Engrs, Trans. Circuit Theory*, **CT-15**, 31

Liu, B. (1971). 'Effect of Finite Word Length on the Accuracy of Digital Filters – A Review.' *Instn elect. electron. Engrs, Trans. Circuit Theory*, **CT-18**, 670

Otnes, R. K. and McNamee, L. P. (1970). 'Instability Thresholds in Digital Filters due to Coefficient Rounding.' *Instn elect. electron. Engrs, Trans. Audio Electroacoustics*, **AU-18**, 456

Weinstein, C. and Oppenheim, A. V. (1969). 'A Comparison of Roundoff Noise in Floating Point and Fixed Point Digital Filter Realisations.' *Proc. Instn elect. electron. Engrs*, **57**, 1181

Recursive filter design methods

Constantinides, A. C. (1967). 'Elliptic Digital Filters.' *Electronics Letters*, **3**, 255

Constantinides, A. G. (1969). 'Digital Filters with Equi-ripple Passbands.' *Instn elect. electron. Engrs, Trans. Circuit Theory*, **CT-16**, 535

Golden, R. M. and Kaiser, J. F. (1964). 'Design of Wideband Sampled-data Filters.' *Bell Syst. tech. J.*, 43, 1533

Lynn, P. A. (1970). 'Economic Linear-phase Recursive Digital Filters.' *Electronics Letters'*, **6**, 143

Shanks, J. L. (1967). 'Recursion Filters for Digital Processing.' *Geophysics*, **32**, 33

Steiglitz, K. (1970). 'Computer-aided Design of Recursive Digital Filter.' *Instn elect. electron. Engrs, Trans. Audio Electroacoustics*, **AU-18**, 123

Thiran, J. P. (1971). 'Recursive Digital Filters with Maximally Flat Group Delay.' *Instn elect. electron. Engrs, Trans. Circuit Theory*, **CT-18**, 659

Non-recursive filter design methods

Gold, B. and Jordan, K. L. (1969). 'A Direct Search Procedure for Designing Finite Duration Impulse Response Filters.' *Instn elect. electron. Engrs, Trans. Audio Electroacoustics*, **AU-17**, 33

Helms, H. D. (1968). 'Nonrecursive Digital Filters: Design Methods for Achieving Specifications on Frequency Response.' *Instn elect. electron. Engrs, Trans. Audio Electroacoustics*, **AU-16**, 336

Herrmann, O. (1970). 'Design of Nonrecursive Digital Filters with Linear Phase.' *Electronics Letters*, **6**, 328

Herrmann, O. (1970). 'Design of Nonrecursive Digital Filters with Minimum Phase.' *Electronics Letters*, **6**, 329

Ormsby, J. F. A. (1961). 'Design of Numerical Filters with Application to Missile Data Processing.' *J. Ass. comput. Mach.*, **8**, 440

Rabiner, L. R. (1971). 'Techniques for Designing Finite-duration Impulse-response Digital Filters.' *Instn elect. electron. Engrs, Trans. Communication Technology*, **COM-19**, 188

Biomedical applications – general

Some of the following papers describe analogue rather than digital implementations. They are included as they may suggest applications for the corresponding digital techniques.

Branston, N. M. and Read, R. R. (1972). 'Catheter Smearing Correction Using Inverse Filtering Techniques.' *Instn elect. electron. Engrs, Trans. Biomed. Engrg.*, **BME-19**, 286

Childers, D. G., Varga, R. S. and Perry, N. W. (1970). 'Composite Signal Decomposition.' *Instn elect. electron. Engrs, Trans. Audio Electroacoustics*, **AU-18**, 471

Fricker, S. J. (1962). 'Narrow band Filter Techniques for the Detection and Measurement of Evoked Responses.' *Electroenceph. clin. Neurophysiol.*, **14**, 411

Jones, R. H. (1971). 'An Adaptive Method for Testing for Change in Digitised Cardiotachometer Data.' *Instn elect. electron. Engrs, Trans. biomed. Engrg.*, **BME-18**, 360 (Gives an example of data reduction by decimation)

Laviron, A. (1971). 'Filtrage Numerique de Rythmes d'Origine Biologique – II.' *Med. biol. Engrg.*, **9**, 109

Smith, J. and Schade, J. P. (1970). 'Computer Programming for Parameter Analysis of the Electroencephalogram.' In *Progress in Brain Research*, Vol. 33: Computers and Brains, ed by J. P. Shade and J. Smith. London; Elsevier

Weaver, C. S., Von Der Groben, J., Mantey, P. E., Toole, J. G., Cole, C. A., Fitzgerald, J. W. and Lawrence, R. W. (1968). 'Digital Filtering with Applications to Electrocardiogram Processing.' *Instn elect. electron. Engrs, Trans. Audio Electroacoustics*, **AU-16**, 350

Wilcock, A. H. and Kirsner, R. L. G. (1969). 'A Digital Filter for Biological Data.' *Med. biol. Engrg.*, **7**, 653

Womack, B. F. (1971). 'The Analysis of Sinus Arrhythmia Using Spectral Analysis and Digital Filtering.' *Instn elect. electron Engrs, Trans. biomed. Engrg.*, **BME-18**, 399

Zetterberg, L. H. (1969). 'Estimation of Parameters for a Linear Difference Equation with Application to EEG Analysis.' *Math. Biosci.*, **5**, 227

Biomedical applications of matched filters

Derbyshire, A. J., Driessen, G. J. and Palmer, C. W. (1967). 'Technical Advances in the Analysis of Single, Acoustically Evoked Potentials.' *Electroenceph. clin. Neurophysiol.*, **22**, 476

Regan, E. (1966). 'An Apparatus for the Correlation of Evoked Potentials and Repetitive Stimuli.' *Med. biol. Engrg.*, **4**, 169

Stark, L. Okajima, M. and Whipple, G. H. (1962). 'Computer Pattern Recognition Techniques: Electrocardiographic Diagnosis.' *Communs. Ass. comput. Mach.*, **5**, 527

Woody, C. D. (1967). 'Characterisation of an Adaptive Filter for the Analysis of Variable Latency Neuroelectric Signals.' *Med. biol. Engrg.*, **5**, 539

Two-dimensional filtering of images

Hunt, B. R. (1970). 'The Inverse Problem of Radiography.' *Math. biosci.*, 8, 161

Hunt. B. R. (1971). 'Block-mode Digital Filtering of Pictures.' *Math. biosci*, **11**, 343. Proc. Instn elect. electron. Engrs, July, 1972, **60**, 766 (special issue on digital picture processing.)

Selzer, R. H. (1968). *Improving Bio-Medical Image Quality with Computers. National Aeronautics and Space Administration;* jet propulsion laboratory technical report. **32**, 1336, Inst. technol. Pacedena, California

Index

Computers in Medicine Series

GENERAL EDITOR: D. W. HILL

THE COMPUTERS IN MEDICINE Series, edited by D. W. Hill, MSc, PhD, FInstP, FIEE, Reader in Medical Physics, Research Department of Anaesthetics, Royal College of Surgeons of England, is a unique collection of specialist monographs dealing with both the clinical applications and the computer science aspects of computing in medicine. This is a fast developing field of medical technology, with wide ramifications in the hospital service, research and general practice. Every branch of the medical profession will value these lucid accounts of computer techniques and applications.

The Butterworth Group 88 Kingsway, London WC2B 6AB